REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
الجمهورية الجزائرية الديمقراطية الشعبية

MINISTRY OF HIGHER EDUCATION
AND SCIENTIFIC RESEARCH

HIGHER SCHOOL IN APPLIED SCIENCES
--T L E M C E N--

وزارة التعليم العالي والبحث العلمي

المدرسة العليا في العلوم التطبيقية
-تلمسان-

Mémoire de fin d'étude

Pour l'obtention du diplôme d'Ingénieur

Filière : Automatique
Spécialité : Automatique

Présenté par : MEKELLECHE Nadjib
BERREHOU Abdelwahab Djacim

Thème

Réalisation d'un robot manipulateur mobile

Soutenu publiquement, le 26 / 09 / 2024, devant le jury composé de :

M. F.ARICHI	MCA	ESSA. Tlemcen	Président
M. B.CHERKI	Professeur	ESSA.Tlemcen	Directeur de mémoire
M. S.M.ABDI	MCB	ESSA.Tlemcen	Examinateur 1
M. H.MEGNAFI	MCA	ESSA.Tlemcen	Examinateur 2
M. B.LASSOUANI	Ingénieur	ALGERIE TELECOM	Partenaire socioéconomique
M. R.A.ADJIM	Ingénieur de laboratoire	ESSA.Tlemcen	Invité 1

Année universitaire : 2023 / 2024

Dedications

I dedicate this modest work to:

My dear parents, whose unwavering love and countless sacrifices have shaped my journey. Your steadfast support and heartfelt prayers have been my guiding light throughout my studies, and for that, I am eternally grateful.

To my younger sister, for her unwavering support and encouragement. May you chase your dreams and achieve great things.

In loving memory of my paternal grandparents, whose wisdom and strength continue to inspire me, and to my maternal grandparents, whose love and guidance have been a foundation in my life.

To my entire family, thank you for your constant encouragement, your sage advice, and the strength you provide. Your belief in me has fueled my ambitions and helped me overcome every challenge.

To all my friends, your support has been invaluable. The bonds we share and the sincere friendships we cultivate enrich my life and motivate me to strive for excellence.

To everyone dear to me, thank you for being part of my journey. Your love and support mean the world to me.

And finally, to you, the reader of this dedication: your presence in my life has been a source of strength. Thank you for standing by me during times of need. Your kindness and encouragement are deeply appreciated.

MEKELLECHE Nadjib

To my mother, who has been with me every step of this journey since the very first day, for the past 17 years. Your love and constant presence have been my source of strength and guidance.

To my father, who has sacrificed so much for me, always ensuring I had what I needed to succeed. Your dedication has not gone unnoticed, and I am forever grateful.

To my brothers, who have been my strength when I needed it most, always standing by me.

To my sisters, whose support and cheers have lifted me up, pushing me to keep going even when things were tough. To my friends, Adel, Zaki, and Wail, my brothers from another mothers, your friendship has meant the world to me.

To that one person reading this dedication, thank you for being there for me when I needed.

BERREHOU Abdelwahab Djacim

Acknowledgments

We express our deep gratitude to Almighty God for guiding us in the accomplishment of this modest work.

We would like to express our gratitude to our families, and especially our parents, for their unconditional support and encouragement throughout our studies.

We sincerely thank **Professor CHERKI Brahim** for his valuable supervision, his enlightened advice and his constant support throughout this project. His guidance was essential to overcome the technical challenges encountered.

We are deeply honored by the participation of **Dr. ARICHI Fayssal**, who chaired our thesis and honored our jury with his presence.

We would like to express our sincere thanks to **Dr. ABDI Sidi Mohammed** and **Dr. MEGNAFI Hicham** for agreeing to review this thesis and for their relevant and constructive comments.

We express our deep gratitude to **Mr. ADJIM Ramz-eddine Abderrezak**, engineer of FABLAB at the Higher School of Applied Sciences of Tlemcen, and **Mr. AISSANI Omar** for their valuable contribution in the realization of our work.

We thank **Mr. Brahim LASSOUANI** for taking the time to evaluate our project and for providing valuable feedback and guidance.

We would also like to warmly thank all our friends as well as all the people who contributed directly or indirectly to the realization of this work. Their support and collaboration were greatly appreciated.

ملخص

يركز المشروع البحثي على تصميم وتطوير نظام روبوتي متحرك مزود بذراع تحكم وعجلات ميكانيكية. يتناول البحث التصميم الميكانيكي واستراتيجيات التحكم، مع تركيز خاص على ضبط وحدات التحكم PID لتحسين دقة الحركة. يشمل المشروع دمج أجهزة الاستشعار مثل الكاميرات ووحدات IMU لتعزيز التنقل والكشف عن الأشياء. النظام المطور يجمع بين الحركة والمناورة البصرية، مما يجعله قابلاً للتكيف مع مهام تتطلب الدقة والمرونة.

Abstract

The research project focuses on the design and development of a mobile robotic system equipped with a manipulator arm and Mecanum wheels. It explores mechanical design and control strategies, with a strong emphasis on fine-tuning PID controllers for precision movement. The project integrates sensors like cameras and IMUs to enhance navigation and object detection. The developed system combines mobility and visual manipulation, making it adaptable for tasks requiring precision and flexibility.

Résumé

Le projet de recherche porte sur la conception et le développement d'un système robotique mobile équipé d'un bras manipulateur et de roues mecanum. Il explore la conception mécanique et les stratégies de contrôle, en mettant l'accent sur le réglage des contrôleurs PID pour une précision accrue des mouvements. Le projet intègre des capteurs comme des caméras et des IMU pour améliorer la navigation et la détection d'objets. Le système développé combine mobilité et manipulation visuelle, le rendant adaptable à des tâches nécessitant précision et flexibilité.

Contents

List of Figures

List of Tables

Abbreviations

- **3D**: Three-Dimensional
- **CAD**: Computer-Aided Design
- **CNC**: Computer Numerical Control
- **USB**: Universal Serial Bus
- **IMU**: Inertial Measurement Unit
- **IR**: Infrared
- **MOSFET**: Metal-Oxide-Semiconductor Field-Effect Transistor
- **DC**: Direct Current
- **IDE**: Integrated Development Environment
- **PWM**: Pulse Width Modulation
- **WIFI**: Wireless Fidelity
- **VNC**: Virtual Network Computing
- **SSH**: Secure Shell
- **HMI**: Human-Machine Interface
- **LiPo**: Lithium polymer
- **VCC**: Voltage Common Collector
- **PID**: Proportional-Integral-Derivative
- **Kp**: Proportional Gain
- **Ki**: Integral Gain
- **Kd**: Derivative Gain
- **RPM**: Revolutions Per Minute
- **IK**: Inverse Kinematics
- **FK**: Forward Kinematics

General Introduction

The field of robotics has seen remarkable growth in recent years, transitioning from simple automation in factories to playing vital roles in sectors like autonomous vehicles, healthcare, and smart homes. The focus of robotics research today is to create systems that can handle complex, adaptive tasks in unpredictable environments. This evolution presents significant challenges, such as processing large volumes of sensor data, making real-time decisions, and achieving reliable, precise performance.

This project investigates these issues by designing, developing, and controlling a robotic system. It examines the entire process, from the initial design to the final control strategies. By addressing both theoretical and practical challenges, the project aims to advance our understanding of robotics potential and its limitations, while also exploring future developments in this exciting field.

The first chapter introduces the broad concepts of robotics, tracing the evolution of robots from simple, automated machines to sophisticated systems capable of performing intelligent tasks. The chapter breaks down the essential components of robots and covers topics such as mobile manipulation, where robots move and interact with objects simultaneously. Different mechanical structures, including robotic arms and mobile platforms, are explored alongside an overview of various types of mobile manipulators, such as multi-legged, underwater, and wheeled robots.

The second chapter highlights the importance of designing robots using 3D models, allowing engineers to simulate their movement and functionality before physical assembly. The chapter outlines the tools and software used to create detailed models of robotic components like the manipulator arm, mobile platform, and mecanum wheels, which give robots omnidirectional movement. These models are essential for visualizing and refining the design from concept to reality.

The third chapter delves into the individual components of the robot, focusing on motors, sensors, and control boards. It provides an in-depth discussion on various motor types, including DC motors, stepper motors, and servo motors, explaining how each contributes to the robots movement. The chapter also covers sensors, such as cameras and line-following sensors, which enable the robot to perceive and respond to its environment. This exploration of components shows how they work together to allow the robot to perform its tasks accurately.

The fourth chapter discusses the practical side of assembling the robot. It outlines the machines and tools used to construct the robotic arm and mobile platform, detailing the process of integrating electronic components with mechanical structures. This chapter

walks through the steps of assembling the robot, addressing challenges such as aligning mechanical parts with the control system to ensure the entire system operates smoothly and efficiently.
The 5th chapter is critical, focusing on how to control the robots movements. This chapter explores the control architecture, detailing how both the mobile platform and manipulator are coordinated. The PID controller is central to this, as it enables smooth, precise movement by continuously adjusting based on sensor feedback. The chapter explains the theory behind PID controllers, how they are tuned for optimal performance, and their limitations. Special attention is given to controlling the yaw angle, a key factor in stabilizing robots with mecanum wheels. Additionally, it explains the use of sensors like the MPU6050 for measuring orientation, and how these measurements are processed to maintain stability. The final section covers the control of servo motors, essential for precise manipulations of the robotic arm, explaining how PID controllers manage both speed and position for accuracy.
The 6th chapter covers camera vision, an essential aspect of modern robotics. Camera systems allow robots to visually perceive their surroundings and make decisions based on what they "see". The chapter introduces the pinhole camera model, explaining how cameras capture images and the mathematics behind it. It also covers camera calibration, ensuring that the robot accurately interprets visual data. Both intrinsic parameters (camera-specific properties) and extrinsic parameters (the cameras position relative to the environment) are discussed, along with how tools like MATLAB are used in the calibration process. This chapter highlights the growing importance of visual feedback in enabling robots to perform tasks such as navigation and object recognition.

Chapter 1

General Overview of Robotics

Robotics is a multidisciplinary field integrating engineering, computer science, and technology to design, build, and operate robots. These machines can perform tasks autonomously or semi-autonomously, often replicating human actions and capabilities.
This chapter provides an overview of the evolution, system composition, mechanical structure, classification, and control aspects of mobile manipulators. Additionally, it delves into the applications of mobile wheeled manipulators, highlighting their significance in various industries and domains.

1.1 Robotics

The Robotics feild is creating machines that are capable of carrying out a variety of tasks, often in environments or situations that are hazardous, repetitive, or beyond human capabilities.
Technological advancements and the growing demand for mobility have led to the rapid development of robots. These robots are becoming increasingly sophisticated, ranging from large stationary machines to small, agile devices that can perform complex tasks and navigate various environments, including factories, offices, and homes. A mobile manipulator is a robotic system that combines a movable platform with a manipulator arm.

1.2 The Evolution of Robotics in the Industry Sector

While productivity, accuracy, repetition and speed have been at focus of research in robotics for years, other criteria based on industrial transformation have become of equal importance [1]. Modern robots must meet criteria such as flexibility, adaptability, precision, and autonomy of action in practically every industry. These characteristics reveal mobile robots and related automation technology play a significant role in the development and advancement of industry. However, it is not only the industrial sector that

benefits from these developments, The Figure (1.1) shows that the world of robotics is increasing in number of installation each year.
Also it is worthy to mention that the supply chain sector too benefits from the new installations and robotic developments.

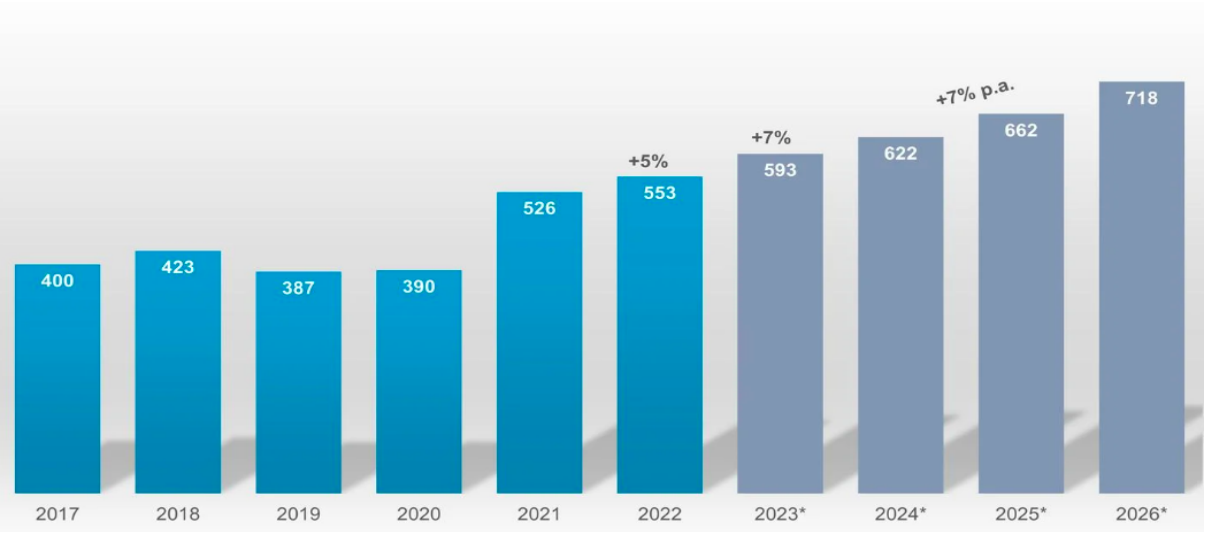


Figure 1.1: Annual Installs Industrial Robots

1.3 General System Composition

Robots can come in all shapes and sizes, but they generally needs the following parts:

- Sensors: These are the robot's eyes and ears, allowing it to gather information about its environment. Sensors can include cameras, microphones, lidar (light detection and ranging) and we can visualize the robot sensors types in the Figure (1.2).

Figure 1.2: Types of Robot Sensors

- Control System (Brain): This is the brain of the robot, where the information from the sensors is processed and used to make decisions about how to move and act. The control system can be a simple circuit board or a complex computer.

- Actuators: These are the muscles of the robot, allowing it to move and interact with the world. Actuators can be motors, hydraulics, or pneumatics.
- Power Supply: This provides the robot with the energy it needs to operate. Power supplies can be batteries, solar panels, or connection to the power grid.

These are the essential parts of a robot, but many robots will also have additional components, such as:

- End Effectors: These are the tools that are attached to the end of a robot's arm or manipulator. End effectors can be grippers, welding tools, spray paint guns, or any other tool that the robot needs to perform its task.
- Locomotion System: This is how the robot moves around. Locomotion systems can include wheels, legs, tracks, or even swimming fins.
- Communication System: This allows the robot to communicate with other robots or computers. Communication systems can be wired or wireless.

1.4 Description of Mobile Manipulation

A mobile manipulation system benefits from the mobility of a mobile platform and the dexterity of the manipulator. The manipulator has increased workspace on the mobile platform. The additional degrees of freedom offered by the mobile platform provide users more options. However, due of the multiple degrees of freedom and the unstructured environment in which it operates, the operation of such a system is difficult.

1.5 Evolution of Mobile Manipulation

Since their inception in 1953, Automated Guided Vehicles (AGVs) have been recognized as autonomous transportation systems. The journey of robotics began with the first industrially programmed robot in 1959, which used coordinate-based programming. In 1972, the development of the first intelligent mobile robot, Shakey, marked a significant milestone. The introduction of IBM's user-friendly robot programming language AML in 1982 further advanced the field. The early 2000s saw the deployment of the Mars rovers by the United States in 2003, showcasing advanced robotic capabilities. The release of the open-source Robot Operating System (ROS) by Willow Garage in 2009 revolutionized robot programming. In 2021, China's development of the Zhurong rover highlighted rapid innovations in robotic technology [2]. These advancements have significantly enhanced robots' intelligence, autonomy, and adaptability, leading to the deployment of Autonomous Mobile Robots (AMRs) in healthcare, robotic vacuum cleaners in households, and intelligent warehousing and social robots. Today, intelligent mobile robots are making significant impacts across various domains, including industry, agriculture, and

defense, marking a new era of intelligence-driven transformation Figure (1.3).

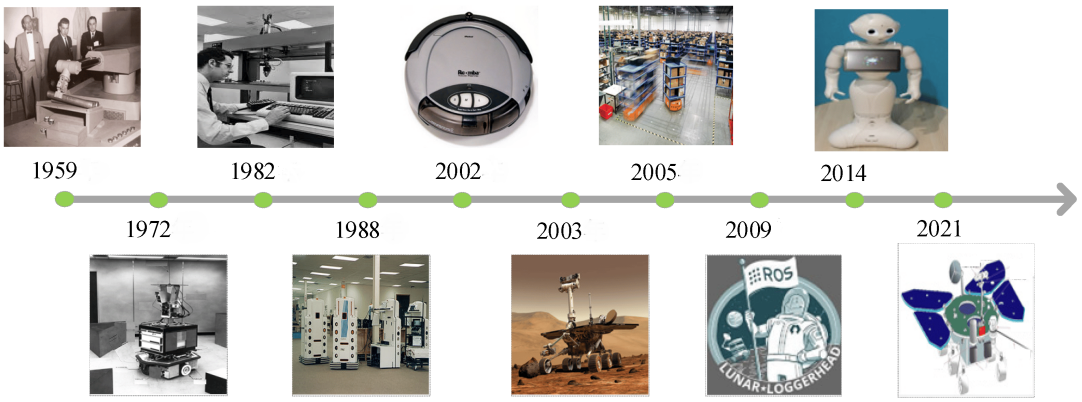


Figure 1.3: Development History of Mobile Robots

1.6 Robot Mechanical Structure

A robot's fundamental aspect is its mechanical configuration. Robots are divided into those with a fixed base, robot manipulators, and those with a mobile base, which are referred to as mobile robots.
The subsequent section details the geometric characteristics of these two categories.

1.6.1 Robot Manipulators

The structure of a robot manipulator involves a sequence of rigid bodies (links) connected through articulations (joints). This manipulator is defined by an arm for mobility, a wrist for dexterity, and an end effector responsible for executing the required task of the robot [3].

Serial Manipulator

A serial manipulator is a type of robotic manipulator consisting of a chain of linked segments. Each segment of the robot is connected to the next, typically by joints that allow rotational or translational movement. The end effector, which is the part of the robot that interacts with the environment, is attached to the last segment of the chain. Types or configurations of serial manipulators include:

- Cartesian Robot: A Cartesian robot operates using a Cartesian coordinate system, where movements are defined in terms of linear motions along the X, Y, and Z axes. These robots are characterized by their simple and precise linear movements, making them well-suited for applications requiring accurate positioning in a fixed workspace Figure (1.4).

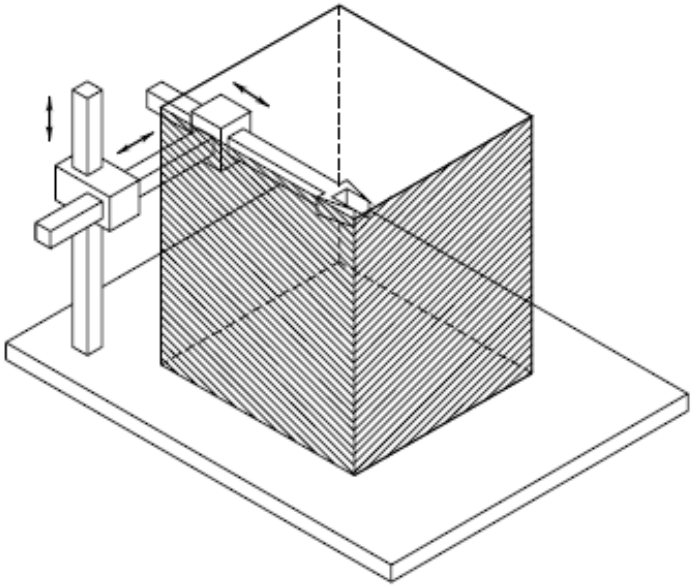


Figure 1.4: Cartesian Manipulator

- Gantry Manipulator: A gantry manipulator is a specialized robotic system designed with a horizontal beam or gantry supported by vertical columns or pillars. This configuration provides a stable and rigid framework that facilitates precise and controlled movement of the robotic arm along a predetermined path within a large workspace. The gantry structure serves as the primary axis for the robot's movement, offering a versatile platform for mounting various end effectors and tools to perform a wide range of tasks Figure (1.5).

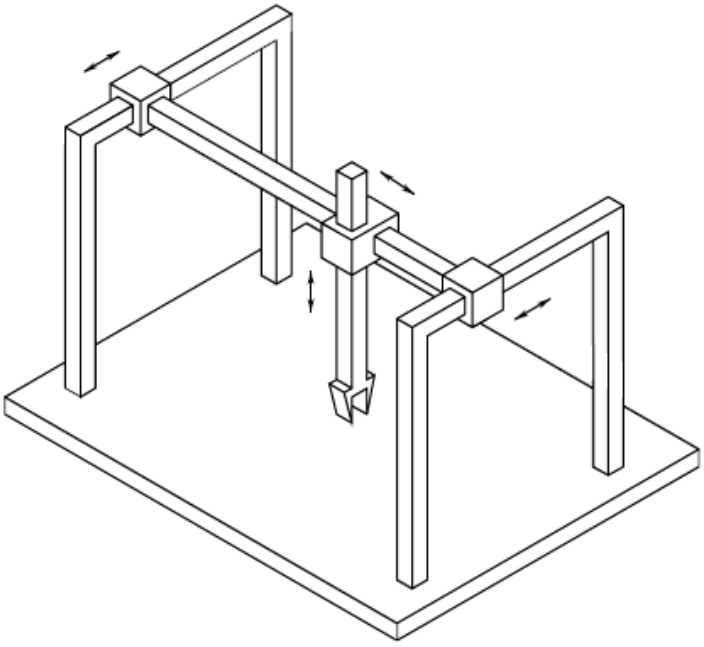


Figure 1.5: Gantry Manipulator

- Cylindrical Robot: cylindrical robot features a combination of rotary and linear

movements. It typically has a rotary joint at the base, allowing rotational movement around a vertical axis, and a prismatic joint for vertical movement. This configuration is advantageous for tasks that involve handling cylindrical or circular objects and requires both rotational and vertical motions Figure (1.6).

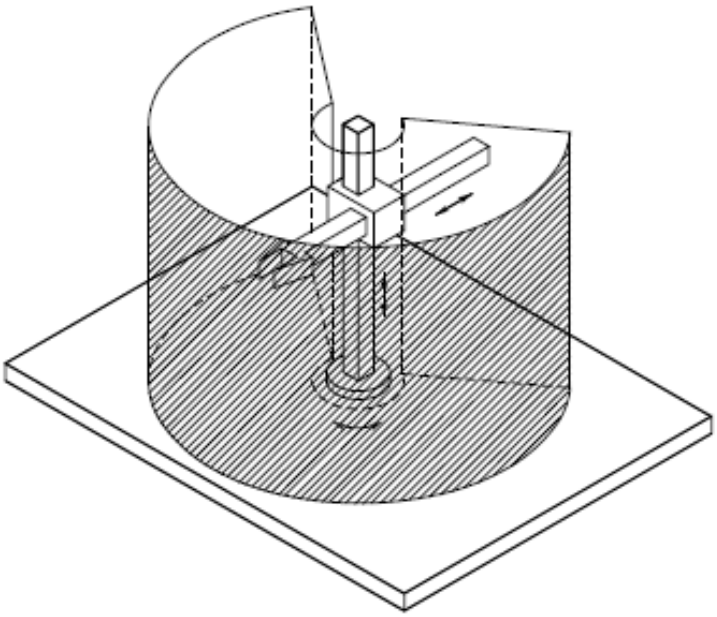


Figure 1.6: Cylindrical Manipulator

- Spherical Robot: A spherical robot operates based on a spherical coordinate system, offering a unique combination of rotational and spherical movements. It has a spherical arm design that enables a wide range of motion and orientation capabilities, making it suitable for applications that require complex spatial movements and positioning Figure (1.7).

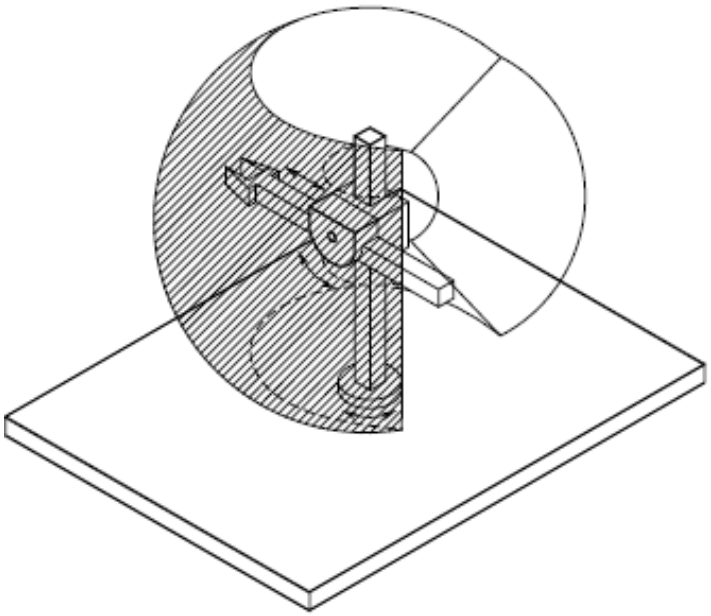


Figure 1.7: Spherical Manipulator

- SCARA Robot (Selective Compliance Assembly Robot Arm):A SCARA robot is characterized by its two parallel rotary joints that provide horizontal movement along the X and Y axes, combined with a prismatic joint for vertical movement along the Z axis. This configuration allows the robot to achieve precise and coordinated movements, making it ideal for assembly tasks, pick-and-place operations, and applications requiring high-speed and high-accuracy positioning Figure (1.8).

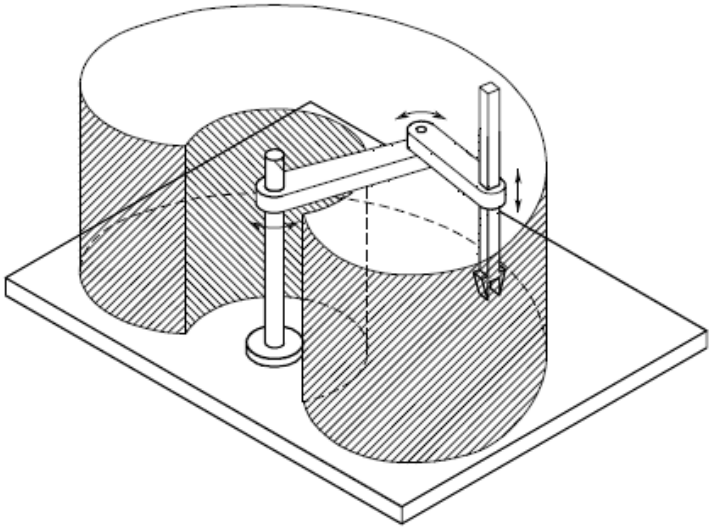


Figure 1.8: SCARA Manipulator

- Anthropomorphic manipulator: An anthropomorphic manipulator, often referred to as a humanoid robot or human-like robot, is a robotic system designed to mimic the structure, function, and capabilities of the human arm and hand. This type

of robot is characterized by its multiple degrees of freedom, allowing it to perform complex and human-like movements with a high degree of flexibility, dexterity, and adaptability. The anthropomorphic design aims to replicate the biomechanics and kinematics of the human body, enabling the robot to interact with its environment in a manner similar to humans Figure (1.9).

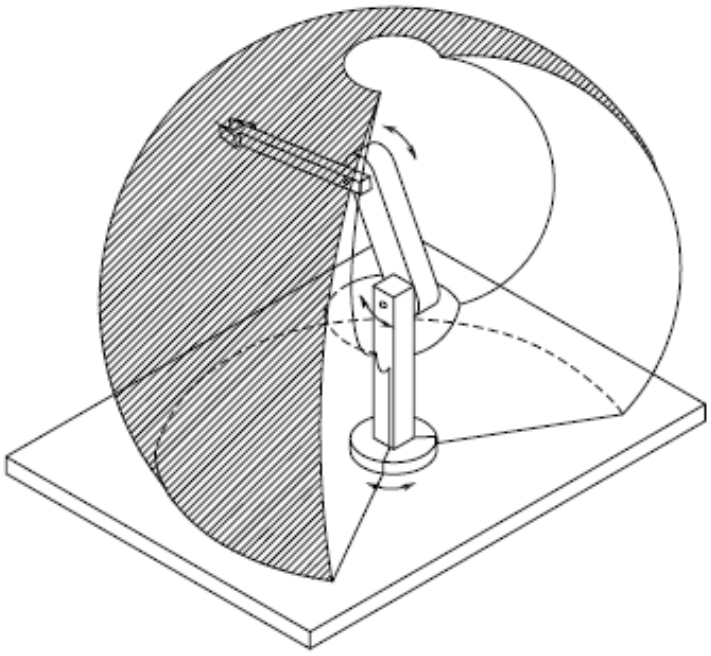


Figure 1.9: Anthropomorphic Manipulator

Parallel Manipulator

A parallel manipulator, also known as a parallel robot or parallel mechanism, is a mechanical system consisting of multiple chains of interconnected links. Unlike serial manipulators, where one link follows another in a linear fashion, parallel manipulators have multiple links connected in parallel to perform motion tasks. These manipulators are widely used in various applications requiring high precision, stiffness, and dynamic performance Figure (1.10).

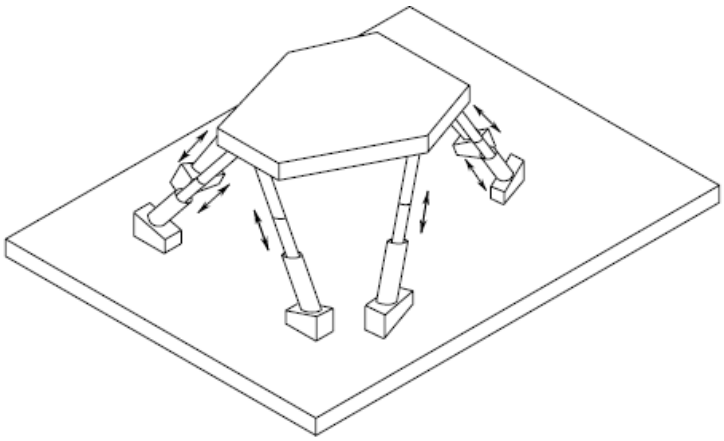


Figure 1.10: Parallel Manipulator

Hybrid Parallel-Serial Manipulator

A hybrid parallel-serial manipulator is an advanced robotic system that merges the structural and operational characteristics of both parallel and serial manipulators. This integration aims to capitalize on the strengths of each manipulator type while minimizing their inherent weaknesses, resulting in a versatile and efficient robotic platform Figure (1.11).

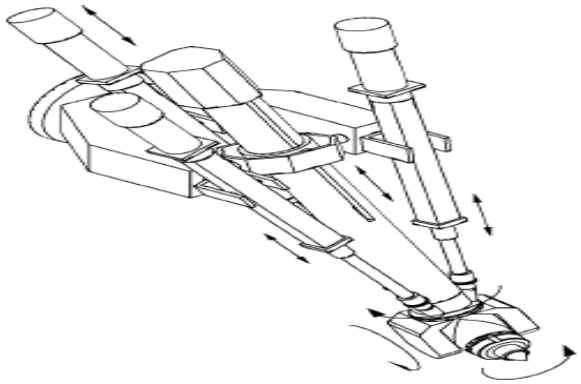
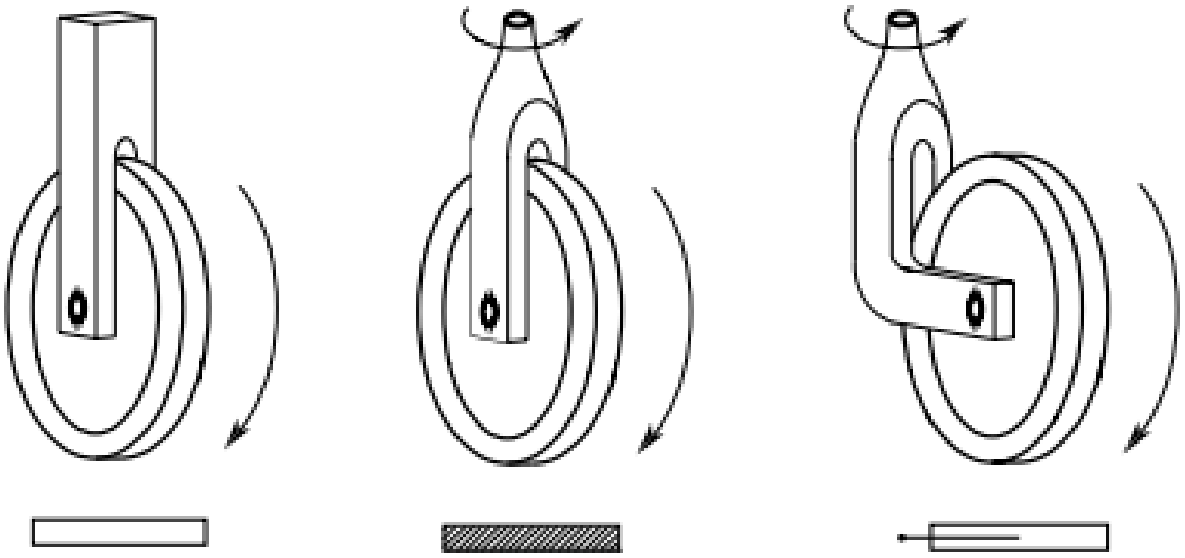


Figure 1.11: Anthropomorphic Manipulator

1.6.2 Mobile Robots

Mobile robots are characterized by having a mobile base that enables them to navigate freely in their surroundings. These robots are commonly utilized in service-oriented tasks that demand advanced autonomous movement abilities. In terms of mechanics, a mobile robot is comprised of one or more rigid bodies that come equipped with a locomotion system.
Wheeled mobile robots generally include a rigid body (base or chassis) and a set of wheels that facilitate movement on the ground. There are three distinct types of conventional wheels Figure (1.12):

- The fixed wheel is capable of rotating around an axis that is perpendicular to the wheel's plane and passes through the center of the wheel. The wheel is firmly affixed to the chassis, resulting in a constant orientation between the two components Figure (1.12a).
- With two axes of rotation, the steerable wheel operates differently. The first axis mimics that of a fixed wheel, while the second axis is vertical and intersects the wheel's center. As a result, the wheel gains the ability to modify its orientation in relation to the chassis Figure (1.12b).
- The caster wheel features two axes of rotation, with the vertical axis positioned away from the center of the wheel by a fixed offset. This configuration enables the wheel to swivel effortlessly and promptly align with the chassis's direction of movement Figure (1.12c).



(a) Fixed Wheel (b) Steerable Wheel (c) Caster Wheel

Figure 1.12: Types of Conventional Wheels

Additionally, in conjunction with the conventional wheels mentioned earlier, there are other specialized wheel variants. Notably, the Mecanum (or Swedish) wheel, illustrated in Figure (1.13), stands out among them. This wheel is fixed and features passive rollers positioned along its outer circumference. Typically, each roller's axis of rotation is inclined at a 45-degree angle with respect to the wheel's plane. When four of these wheels are mounted in pairs on two parallel axes, the vehicle gains omnidirectional capabilities.

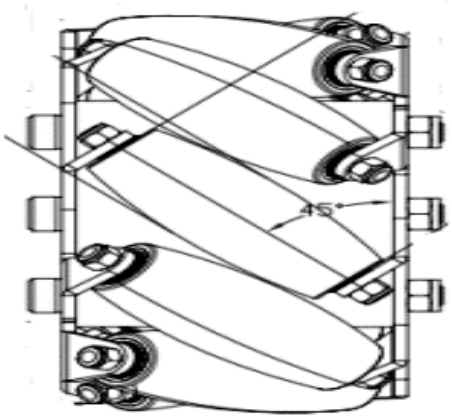


Figure 1.13: Mecanum Wheel

1.7 Classification of Mobile Manipulator

The concept of the mobile manipulation consists of associating, in the same system, one or more means of locomotion with one or more means of manipulation. The means of locomotion provide the system with a working space limited mainly by its energy autonomy. The objective of manipulation is ensuring that the system is able to move and manipulate objects.
Among the systems that currently exist we can mention three main families.

1.7.1 Multiped

As the name suggests, multi-legged mobile manipulators can have one or more legs. Humanoids are the most popular representatives of this type of manipulator as it can be seen in Figure (1.14), both with the general public and with researchers, because of the challenge they represent. From the point of view of manipulation, their possible uses are limited from an industrial point of view and their main outlet is service robotics.

(a) Quadruped Robot (b) Hexapod Robots

Figure 1.14: Multiped Robot

1.7.2 Underwater Mobile Manipulators

Underwater mobile manipulators are nowadays the most widely used mobile manipulators in reseach field for work purposes. Often remotely operated, they are called Remotely Operated Vehicles (ROV) and allow access to maritime areas not accessible to these provide sampling, manipulation, measurement and data acquisition capabilities that can be adapted to the missions required missions. An example is the Victor 6000 as it can be seen in Figure (1.15)

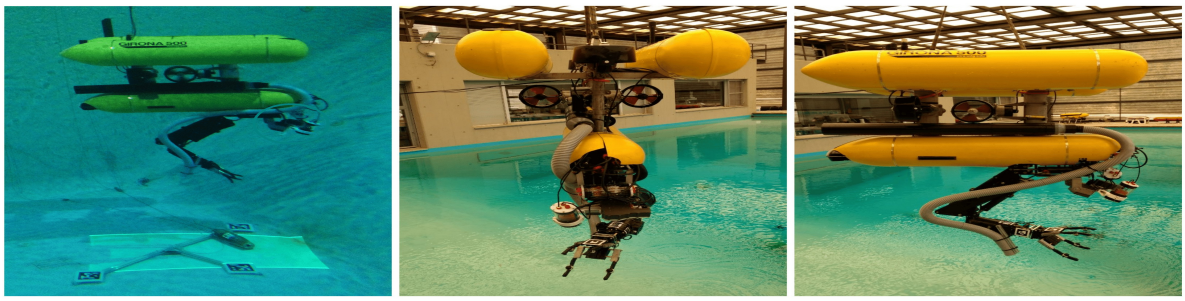


Figure 1.15: Underwater Mobile Manipulators

1.7.3 Mobile Wheel Manipulators

Wheeled mobile manipulators Figure (1.16) are more common in the industrial field than those presented previously. This is due particularly due to two facts:

- Their relatively simple mechanical design.
- The natural suitability of their means of locomotion to a wide range of terrains.

Figure 1.16: Mobile Wheel Manipulators

1.8 Control of Mobile Manipulators

Control of mobile manipulators involves coordinating the motion and manipulation capabilities of a robotic system that combines both mobility (movement) and manipulation (grasping, lifting, etc.) functionalities.

1.8.1 Control of the Manipulator Arms

The objective of controlling the manipulator arms is to control the joints that constitute them so that they follow the predefined paths to reach the final position while respecting the kinematic constraints linked to the movement.

1.8.2 Control of Mobile Robots

On the mobile robot side, motion control and trajectory tracking are essential for navigating the robot through the environment while avoiding obstacles.

1.8.3 Controlling Both of them

Controlling a mobile manipulator requires management of both mechanical systems designed in a distinct way and reacting differently to external influences. In this type of platform the notion of redundancy appears, which leads to an infinity of system configurations for a given situation, leading to the design of modes of innovative controls.

1.9 Applications of Mobile Wheeled Manipulators

Mobile wheeled manipulators can be used in different areas of applications. They are divided into three categories : logistics applications, support and service.

- Warehousing and Logistics: In warehousing and logistics, these robots can assist in picking and packing operations, inventory management, and goods transportation.
- Manufacturing: Mobile manipulators can be used in manufacturing environments for tasks such as picking and placing objects, assembly tasks, and machine tending.
- Healthcare: Mobile manipulators can be employed in hospitals and clinics for tasks like patient assistance, delivery of supplies, and handling medical equipment.
- Agriculture: In agriculture, these robots can be used for tasks such as harvesting crops, planting seeds, and monitoring crops for pests and diseases.
- Search and Rescue: Mobile manipulators equipped with sensors and cameras can be deployed in search and rescue missions to navigate through complex terrains and manipulate objects to assist in rescue operations.
- Education and Training: Mobile manipulators are used in educational settings to teach robotics concepts, programming, and automation skills to students.

Chapter 2

System Design

For building a mechanical structure of a robot, the 3D design is critical step, this last allows designing and setting the compatible dimensions for the manipulator robot depending on the available materials, also helps in the visualization,the simulation and the construction. This chapter presents the used designs in the robot manipulator conception.
We have created a mobile robot manipulator design and modified a 3D model of a 'Braccio TinkerKit' robotic arm which is an open-source file via the design software and 3D modeling SolidWorks.

2.1 3D Model

2.1.1 Description

A 3D model is a digital representation of the real robot in three dimensions. The virtual environment allows us to integrate details , components and change colors or texture. After defining the characteristics we can simulate the robot or the object also visualize it or move it.

2.1.2 Designing Software

To design our prototype we used a dedicated software called SolidWorks, it is a high performance software and it's well known in the engineering community for its capabilities to create any imaginable 3D model. It is a computer-aided design (CAD) and engineering (CAE) used for such fields as mechanics, electronics, architecture, civil engineering and many others. This last helps to design, create, edit, simulate and explore 3D models.
In robot designing, SolidWorks and other softwares such as Fusion360 provide an ideal platform which facilitate the process of creating detailed 3D models and assemble the different parts to be used for the planning and the simulations.

2.2 Presentation of the 3D Model

The 3D modeling step consist of creating a sophisticated combination of robotics components using computer graphics. This digital model represents a mobile robot equipped with a manipulator arm, which is designed with precision to perform high accuracy movements. It facilitates the 3D printing, and the understanding of the robot kinematics, dynamics, and control strategies. This approach reduces time and costs, This step can help also to predict mechanical problems and in other words "prevention".
We could navigate the robot through a virtual environment and to interact with objects through the robotic arm.

2.3 Robotic Arm 3D Model

The arm structure needs to be sturdy and durable to handle heavy use but the joints are flexible and precise to mimic the human movements and interact with delicate objects.
The manipulator arm is built of a multiple joints and links calculated to give the robot the flexibility to reach and catch objects around the whole robot.
Each movement is a result of a rotation of a servo motor rotating the attached link to its gear, the motion is controlled via Arduino micro controller.
a detailed view of the internal structure of the arm in the following figure allows a better understanding of its functionality.
TinkerKit Braccio is a 5 degrees of freedom robotic arm, equipped with 6 rotating joints and 3 links followed with an utility head known as (the gripper) [4]. The original arm 3D model is equipped with SR 431 and SR 311 type servo motors Figure (2.1).

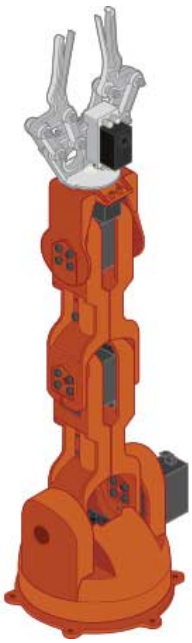


Figure 2.1: Braccio Robot Equipped with SR311 and SR431

In the implementation phase, we were obliged to deal with the available servo motors which are in this case the MG996 and SG90 type. We have modified the model using SolidWorks in order to the make a placement for the MG996 and SG90 servo motors Figure (2.2).

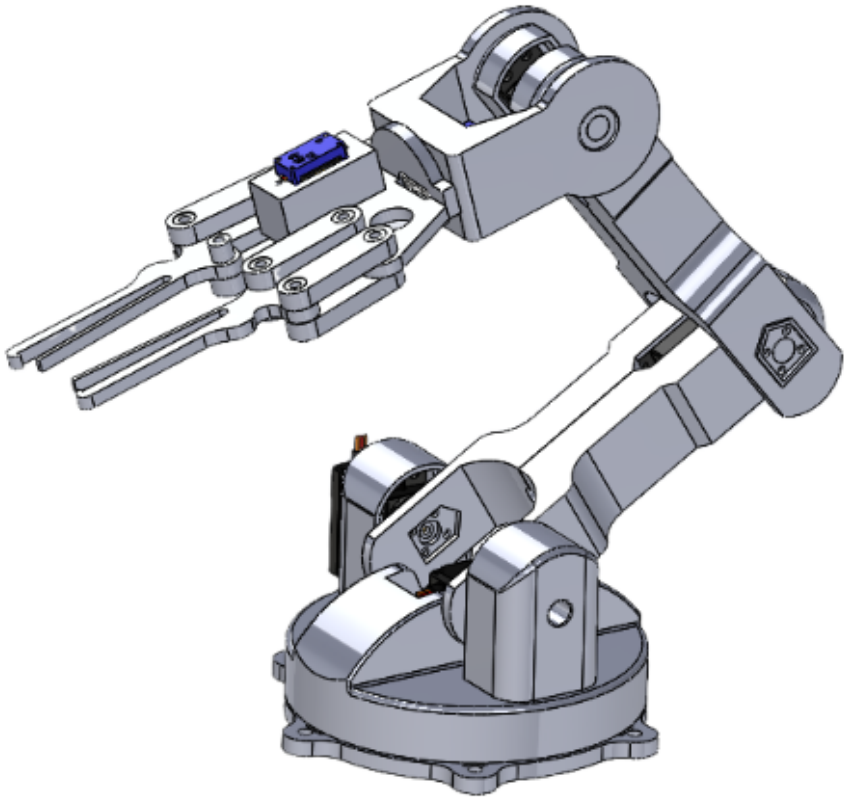
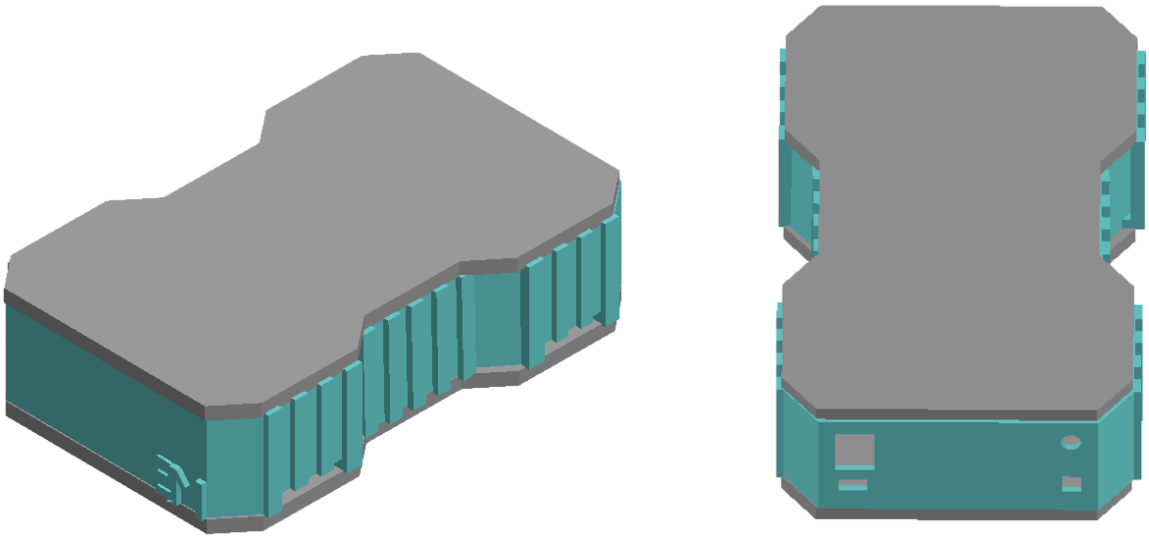


Figure 2.2: Robot Manipulator Equipped with MG996 and SG90

2.4 Mobile Robot 3D Model

We have chosen to design our own mobile robot which Responds to the requirements and the conditions such as the desired dimensions to hold the robotic arm and the different components including the large ones for example (the lithium battery, the power bank, L298N driver and the DC motors).
This model is created to merge between the functionality and the look, this combination is achievable with right chose of dimensions, the number of electrical wires inputs and their placements, the fixation holes and the different creative esthetic touches.
The following Figures (2.3) and (2.4) represent a well detailed view of all the 3D chassis model from the main sides.



(a) Side View (b) Back View

Figure 2.3: 3D Model Representation of the Robot Mobile Chassis

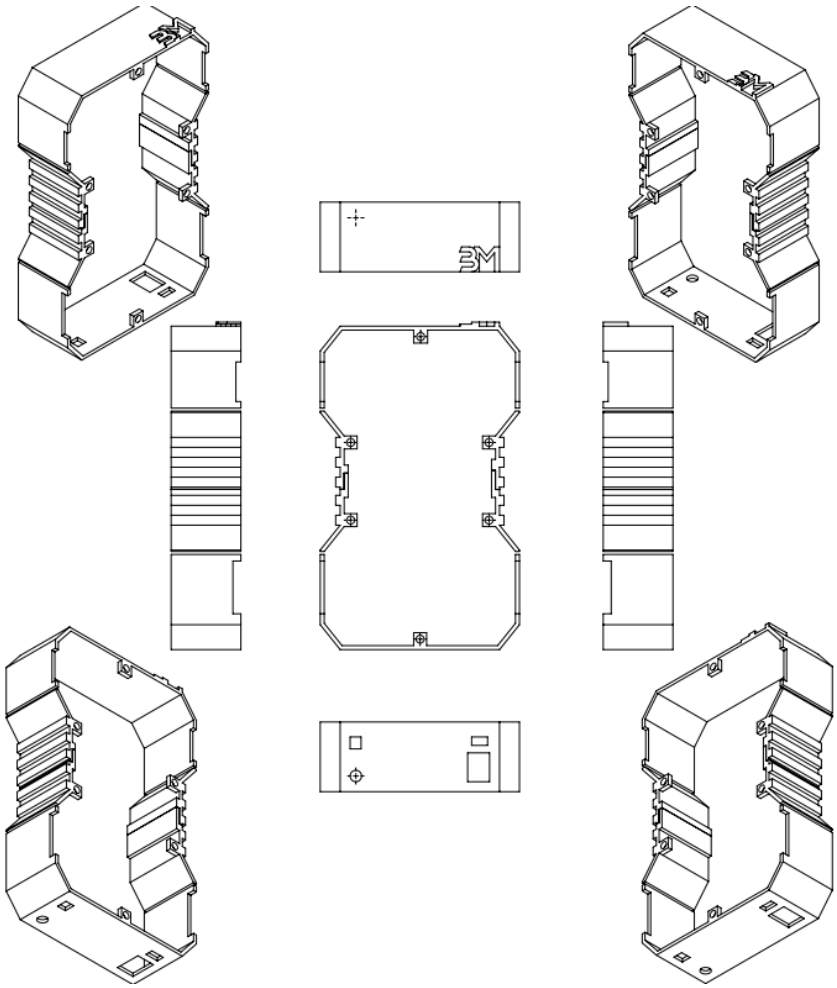


Figure 2.4: Descriptive Drawing of the Robot Mobile Chassis

2.5 Mecanum Wheels 3D Model

The mecanum wheels are well designed to perform a high precision translation movements for the mobile robot in the (X;Y) axes panel using rotational movements, this kind of translation is achievable using 10 rollers attached to each wheel in a 45 degrees position and they are all motorized using one motor coupled to the centre of the main wheel which results a fraction force with the ground tangential to the roller.
The rollers are not cylindrical but they have a curved contact surface to make the interaction with the ground smooth and to reduce noice and instability.
the mecanum wheels are not identical but there is two different types (left and right) placed diagonally, each type is fabricated in a separate model.
In the following Figure (2.5) we can see the details 3D model of the mecanum wheels [5].

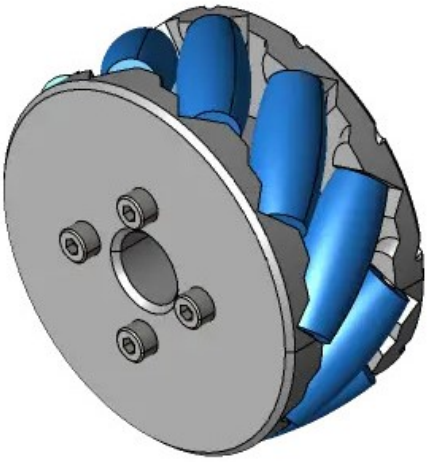


Figure 2.5: Mecanum Wheel Assembled 3D Model

For a better view we can explore the separated parts of the mecanum wheels in the next Figure (2.6).

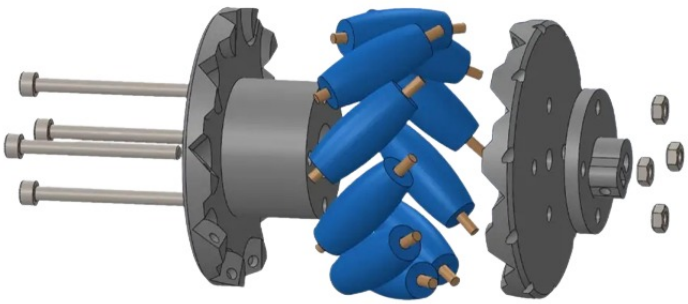


Figure 2.6: Mecanum Wheel 3D Model Separated

2.6 Gimbal 3D Model

The gimbal is a mini robot used to allow the cameras or the ultra sonic sensors to rotate and expand their field of visualisation, it is equipped usually with two axes of rotation to achieve rotation around the horizontal and the vertical axes, we have designed a gimbal to control the camera angle separately of the main robot using a servo motor which rotates around the vertical axis, and we could extract this gimbal transformation matrix in the camera vision chapter.
The Figure (2.7) provide us with the 3D Model for ensure a better understanding.

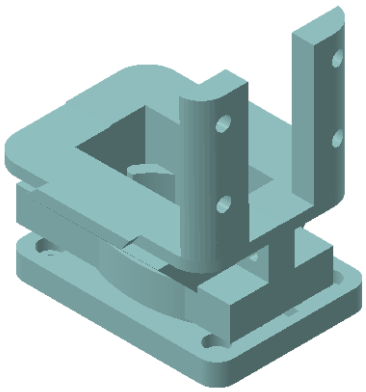
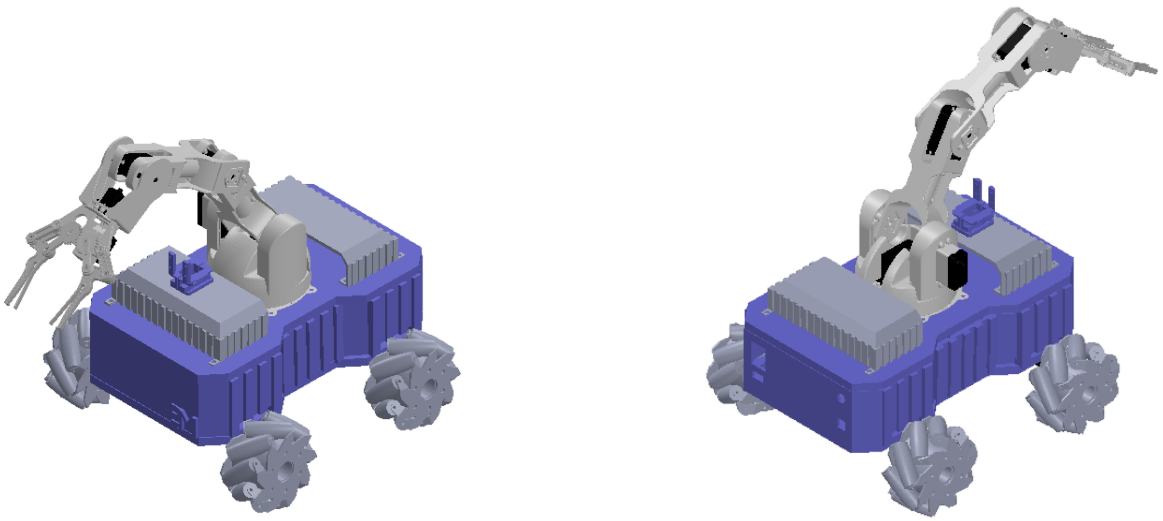


Figure 2.7: 3D Model of the Fully Assembled Gimbal

2.7 3D Model Assembly of Mobile Robot Manipulators

The previous parts are assembled respecting the dimensions and the weights distribution constraints using distributed places of fixation axes along the arm and the twelve chassis holders to the base, and this design couldn't be achievable without adding some parts including the dc motors supports and rotation axes added to the mecanum wheels and the arm gripper, and it is necessary to mention the holding mediator between the arms links and the servo motor gear.
Since we did not ignore the esthetic aspects, we added two covers on top of the mobile robot chassis to hide the raspberry pi controller and the numerous wires heading towards the Arduino Mega board.
To help visualizing the assembled mobile robot manipulator and different added parts we have attached the Figures (2.8) and (2.9).



(a) Front Right View (b) Back Left View

Figure 2.8: 3D Model Representation of the Assembled Mobile Robot Manipulator

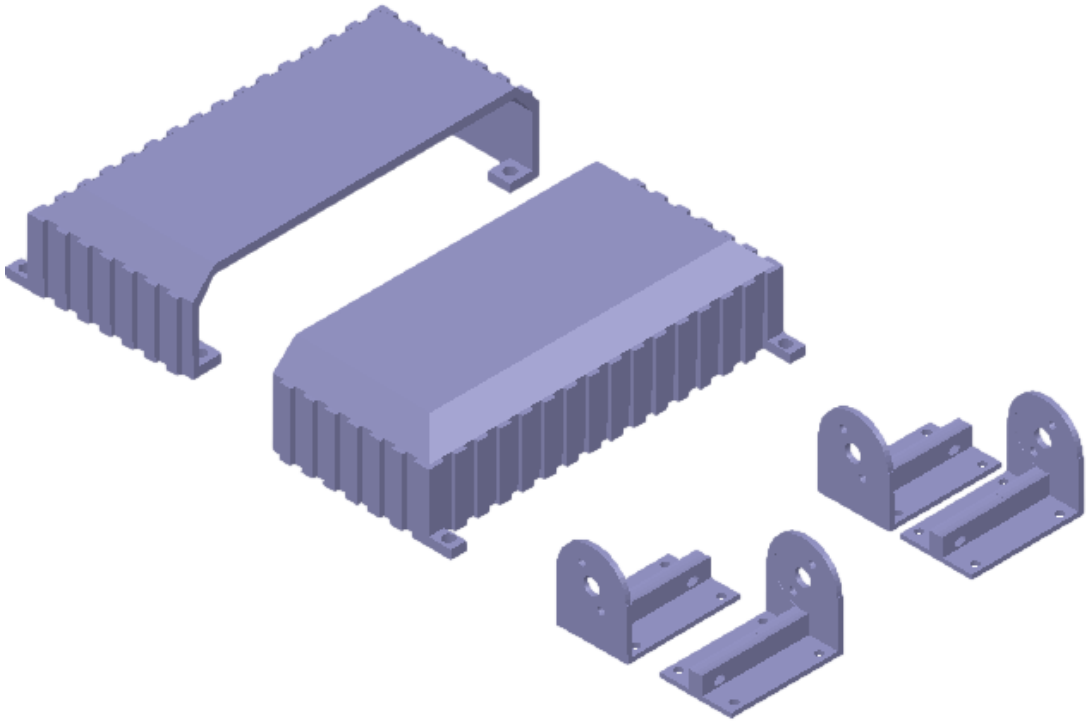


Figure 2.9: 3D Model of the Cover and DC Motor Supports

Chapter 3

Components of Robotics Systems

The manipulator robot is a result of the interaction between the mechanical the electronic parts including motors, control boards, batteries and power boards and the action depends on the communication and the compatibility of the components.
In this chapter we are willing to present all the components used in this project mentioning the causes of use, the advantages and the disadvantages of each component.

3.1 Overview of Motors

Motors are devices that convert electrical energy to a mechanical rotation around a stationary axis and the first electric magnetic modern machine was built in 1832 by William Sturgeon.
These devices are generating force and torque for multiple applications starting from the industrial uses and engineering until the hobbyists, including 3D printers, electric vehicles, robotics, elevators, conveyor systems, security cameras. it is necessary to understand their functionality, the difference between each motor, their applications and the necessary drivers to control it to chose the best option.

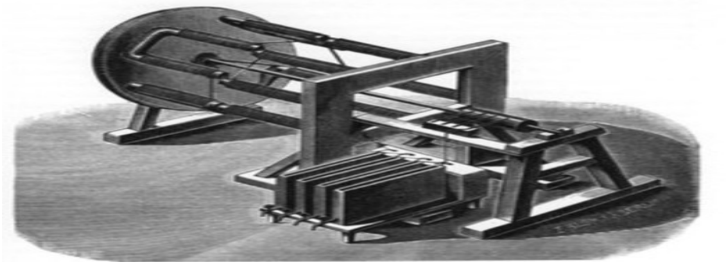


Figure 3.1: Principles of the Sturgeon Motor

3.1.1 Direct Current Motor

A dc motor is an electrical machine provide mechanical energy in a rotation form by creating a magnetic field in its stator using the direct current which is going to attract and rotate the magnets on the rotor and to keep this rotation continues we use an attached commutation to the brushes connected with the current source. DC motors are known for their ability to control the output speed and torque by changing the input voltage and current and their variety of sizes depending on the utility needs adding to that the simplicity of installation and finally the low cost the availability.

3.1.2 Stepper Motor

Stepper motors are a DC motors that move in discrete steps. They have multiple coils that are organized in groups called "phases". By energizing each phase in sequence, the motor will rotate, one step at a time.
A stepper motor is a brushless, synchronous electric motor that converts digital pulses produced by the control board into mechanical shaft rotation. The shaft motion is achievable by accumulating the discrete angular movements resulted from sequentially switching the phase energized.

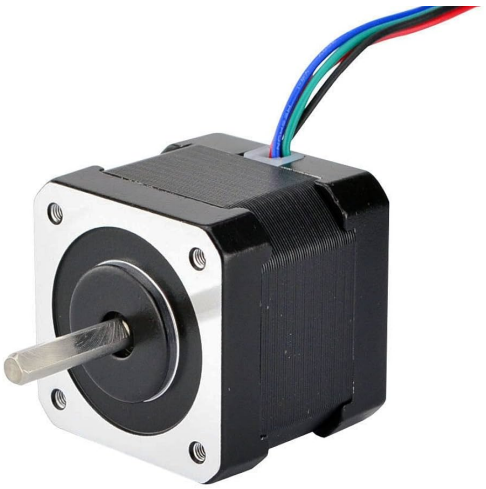


Figure 3.2: NEMA17 Stepper Motor

Justification

The stepper motors are known for their high precision steps for distances crossing and its high torque even at the maximum speed but we need to mention that it uses a delay function to moves controlling a phase and stopping the other at same time which is going to forbid us from a real time data treatment and built in sensors using to get the position feedback and calculate the speed, we can not ignore the fact of their high mass value which is going to consume more current and power and using the forward command by manipulating the set point we can't achieve a full control on the robot or its variables.

3.1.3 JGA25 370 Motor with Encoder

The JGA25-370 DC12V 915RPM Motor with Encoder is a direct current (DC) electric motor that operates at a voltage of 12 volts. Its maximum rotation speed is 915 revolutions per minute.
It is equipped with an encoder, a device that converts mechanical movement into electrical signals to enable precise measurement of motor speed, direction and position. This type of motor is widely used in applications requiring precise motion control, such as mobile robots, drones, autonomous vehicles.

Figure 3.3: JGA25 370 DC Motor

Characteristics

Working Voltage	6 V to 18 V
Nominal Motor Voltage	12 V
No-Load Speed at 12 V	915 RPM
No-Load Current at 12 V	50 mA
Stall Current at 12 V	1200 mA
Stall Torque at 12 V	1 kg.cm
Gear Ratio	1:9.6

Table 3.1: JGA25 370 915RPM Motor Characteristics

Justification

The encoders, integrated into the motors of the mobile robot, measure the position, speed and direction of rotation. They convert movements into electrical signals, allowing precise closed-loop control. Incremental encoders track position variations, while absolute encoders provide the exact angular position. They ensure precise and reliable movements of the robot. They also provide essential information to correct movement errors in real time, ensuring efficient and accurate navigation [6].

3.1.4 Servo Motor

A servo motor is a rotary actuator that allows a precise control of angular position which is needed for precise angles achievement such as robotic arms. It consists of a motor coupled to a sensor for position feedback. It also requires a servo controller to complete the system which uses the feedback sensor to precisely control the rotary position of the motor [7].

Figure 3.4: MG996R Servo Motor

Servo Motors Working Principle

Servo motors consist of four main components which are: the gear assembly, the position sensor, the DC motor and the control circuit. The voltage variation changes the output speed then the error of position is detected by a potentiometer attached to the output gear which represents the servo motor shaft current position and the potentiometer output enters an error amplifier. In order to represent the desired setpoint angle or speed we must apply a reference signal which is in a pulse width form then it gets transformed into a direct current voltage value and each voltage value is appropriate to an angle degree. The internal controller compares the setpoint value to the output value then adjust the output Driving the motor until eliminating the error. If the error is negative the controller changes the motor direction without causing a full turn problem.

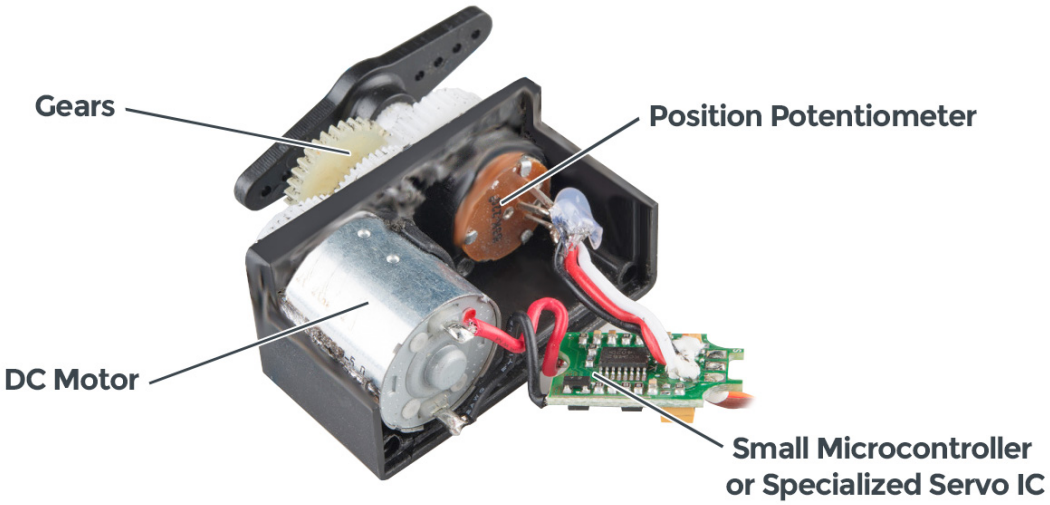


Figure 3.5: Conceptual Design of the Servo Motor

Characteristics and Features

The MG996R is the used servo motor for the robotic arm motion and it is an upgraded version of the Towerpro MG995 servo and it has some features such as:

- PCB and IC control system for a better respond time and less static error.
- The gear box and motor are centered to improve the dead bandwidth zone.

Dimension	40mm x 19mm x 43mm
Weight	55g
Operating Speed (4.8V no load)	0.17sec / 60 degrees
Operating Speed (6.0V no load)	0.13sec / 60 degrees
Stall Torque at 4.8V	13 kg-cm
Stall Torque at 6V	15 kg-cm
Operation Voltage	4.8V - 7.2V
Gear Type	Metal Gears
Connector Wire	300mm

Table 3.2: MG996R Servo Motor Characteristics and Features

Modified Servos

Modifying the servo motor is an operation consist of soldering a wire with the point connected to the signal output of the potentiometer attached to the servo motor last gear, This process will allow us to get the angular position and the speed feedback to test the internal regulation loop performances including respond time and static error and even identifying the transfer function of the servo motor and its controller [8].

Justification

If we chose to use a DC motor we need to deal with the position and the speed control which is no attainable without a gearbox, a feedback and a controller which are provided by the servo motor. for any kind of motors in order to generate high torque values we need to supply a high voltage value which causes the increase of the motor speed forbidden us from a precise control and the low voltage causes the torque decrease.
This problem can be solved with the servo motors or the stepper motors which are both widely used in the robotic arms field, and here are a comparison that explain the taken choice of the servo motors.

Servo Motor	Stepper Motor
operates in a closed-loop	operates in an open-loop
internal feedback system and less error	no feedback system and more error
cheap (in the case of MG996R)	expensive
small (in the case of MG996R)	bigger then the MG996R
lower torque at low speeds	high torque
high torque at higher speeds	low torque
require an encoder and a gearbox	no need
The speed is higher	lower
pulsate or vibrate in the standstill position	no vibration or pulsation

Table 3.3: Comparison Between Servo Motors and Stepper Motors

We have chose the MG996R precisely among the other servo motors because unlike the other models it has a metal gears set instead of plastic fragile once and these gears are the provider of the high torque values.

3.2 Drivers

3.2.1 A4988

The A4988 is a driver used to control bipolar stepper motors speed and micro stepping, This last can support maximum voltage and current up to of 35V and 2A with a built in current regulator, The driver provide us with multiple step modes for a simple control in case of lacking of a complex microprocessor and it can command each phase separately using the pins (A1, A2, B1 ,B2) which gives it the ability and change the direction
In another hand, This driver is the available option which can improve step accuracy reducing motor noice and power dissipation

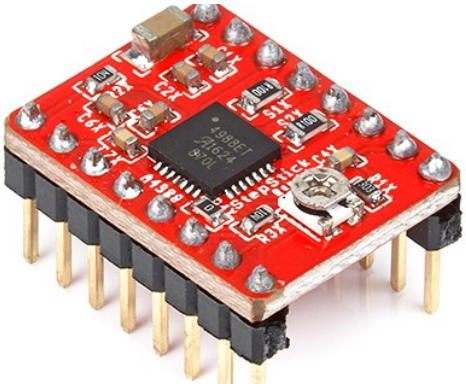


Figure 3.6: A4988 Stepper Motor Driver

3.2.2 L298N Double Driver

L298N Dual H-Bridge Motor Controller, typically used to control motor speed and rotation direction. It can also be used for other products such as with LED arrays, relays, and solenoids, etc. It's a powerful little motors driver.
It is equipped with a heat sink in order to keep the operational temperature of the driver which is capable of powering 5-35V motors with a max of 2A by (0V up to 5V) to an output power supply for the motors (0V up to the maximum voltage supply value)

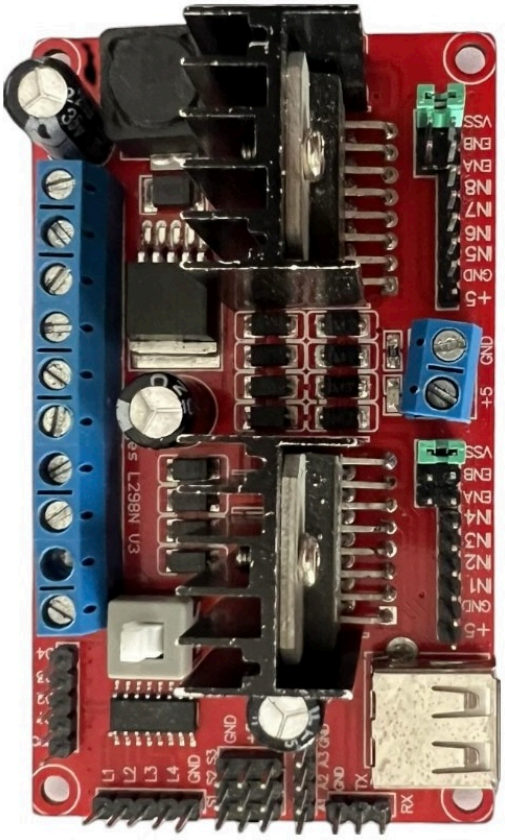


Figure 3.7: L298N Motor Driver

3.2.3 DRV8833

The DRV8833 is an optimized motor control driver which is equipped with two NMOS H-bridge drivers, this last can control inductive loads such as a bipolar stepper motor or DC motors even a solenoid. The driver operates between 2.7V and 10.8V but after the testes using the DC supply source, we could use 12 Volts as an operation voltage without an issue and this driver is able to continuously supply 1.2A in each channel and it can resist a peak current of 2A as the limit channel current and this driver includes two channels each one of them can drive and control a DC motor [9].
The driver offers some protection capabilities including the under-voltage lockout, the over-current and the over-temperature protections offering a safe an efficient module to use in the robot manipulator. Any of these past protections turn off the drivers MOSFETs, and after the elimination of the fault the driver continue the control. We also mention that the DRV8833 has a sleep mode which saves the energy and reduce the battery consumption time.

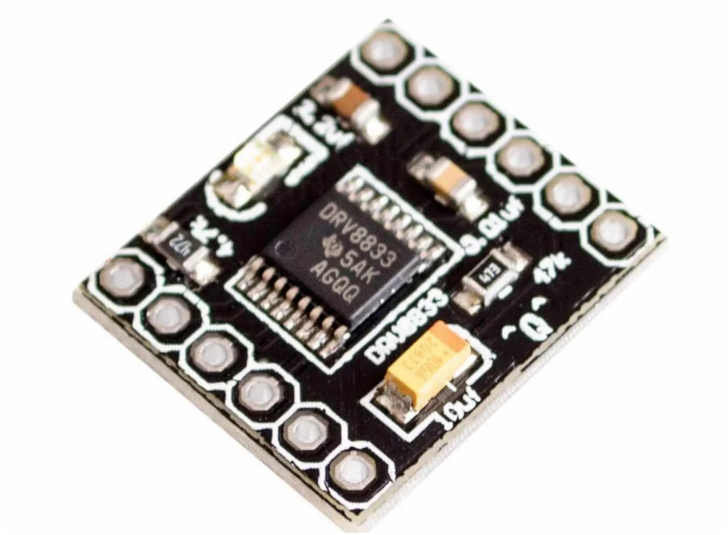


Figure 3.8: DRV8833 Motor Driver

Justification

The L298N and the L293D are the most used motor drivers because their simplicity and availability but their disadvantage is that these past drivers use a bipolar junction transistors BJTs, which makes them less efficient and reliable because in the case of voltage drop the BJT enters the ON-state and starts consuming energy which is needs to be dissipated as heat usually causing and overheat and an unnecessary energy consumption. In the other hand the MOSFETs has a negligible voltage drop leading reduce the temperature and the energy consumption.

3.3 Cameras and Sensors

3.3.1 Raspberry Pi V2 Camera Module

The V2 camera Module of the Raspberry pi company is a special small camera developed for the raspberry Pi boards applications with characteristics such as:

- Resolution: It has an 8-megapixel Sony IMX219 sensor which delivers high quality images with a resolution of 3280 x 2464 pixels.
- Video Capability: It has the option of high quality recording or high speed captures for different applications (1080p at 30 frames per second, 720p at 60 fps or 640x480p at 90 fps).
- Lens: It has no option to add an external lenses but it is equipped with a fixed focus lens with a focal length of 3.04mm and an aperture of f/2.0 and an horizontal view of 62.2 degrees.
- Connectivity: The connection is provided via a dedicated camera serial interface (CSI) port on the Raspberry Pi and a cable a 15 pins ribbon cable for direct communication.
- Compatibility: It is compatible with all the Raspberry Pi boards models which includes a camera port.
- Size: designed for development projects provides a small size of 25mm x 23mm x 9mm.
- Software Support: It is supported by Raspbian operating system using the raspistill or raspivid command line for capturing or recording, it also can be used by the programming libraries like OpenCV or Picamera.

Applications

The camera used for the robotics projects, prototyping, security systems and embedded system applications.

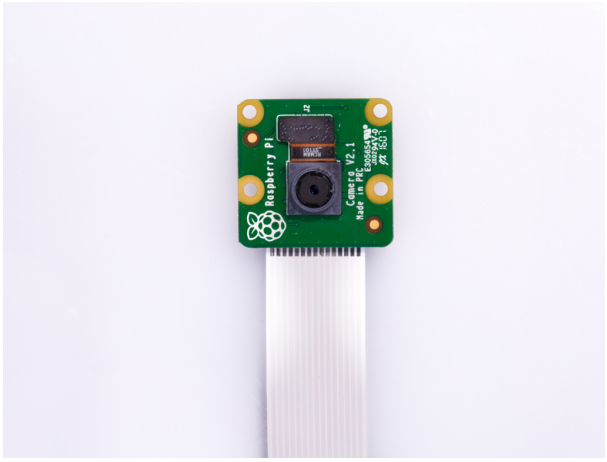


Figure 3.9: Raspberry Pi camera V2

3.3.2 MPU 6050

In robotic applications or any movement control we need a real 3D visualization or image of the robot state using the MPU sensor orientation. In our project we are interested in using the MPU6050 in order to measure angular position of the robot mobile to ensure the position reach with precision, its reachable by fusing the accelerometer and the gyroscope data to get reliable values. The MPU6050 is an Inertial Measurement Unit (IMU), it is an electronic sensor that gives feedback of the acceleration, the angular velocity, the temperature. This IMU is a 6DOF (degrees of freedom) device includes 3-axis accelerometer and 3-axis gyroscope for each axe X, Y and Z.
The accelerometer and the gyroscope measure the acceleration and the rotational velocity on the 3-axis using MEMS technology and Coriolis effect.
The MPU6050 is a low-cost high efficiency solution for measuring angular position or velocity the same as our case, it uses the I2C communication protocol to transfer measurements to the Arduino using two pins only (SDA and SCL) which represents the data line and their time line, and low supply voltage 3V-5V and GND.
The IMU measures can be approved by fusing and combining the two measurements of the accelerometer and the gyroscope to decrease the errors and also adding a Bandwidth Filter helps smoothing out the signal by modifying the cutoff frequency of the low pass filter which leads the remove high frequency noise [10].

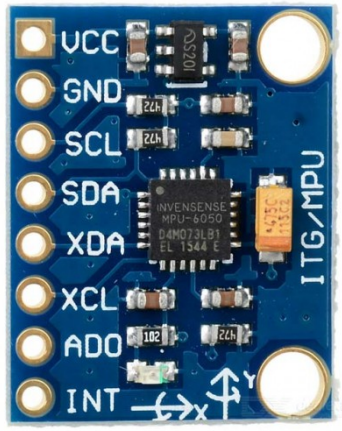


Figure 3.10: MPU6050 6-Axis Gyroscope and Accelerometer Module

3.3.3 Line-Following Sensor

A line-following sensor is a key component in robotics, enabling a robot to detect and follow a line on the ground, typically a dark line on a light surface or vice versa. In the case of the 5-channel line-following sensor, it consists of five infrared (IR) sensors aligned in a row. These sensors work by detecting the reflectivity of the surface beneath the robot [11] [12].

Operation of the Sensor

- **Detection**: Each IR sensor emits an infrared beam and measures the reflected light. A dark line reflects less light than a lighter surface. When the sensor passes over the line, it detects the low reflection, and when over a lighter surface, it detects higher reflection levels.
- **Five Channels**: The five individual sensors provide multiple detection points. Depending on which sensor detects the line, the robot's control system can determine the line's position relative to the sensor array.
- **Positioning**: If the central sensor detects the line, the robot is properly aligned. If the line is detected by the left or right sensors, the robot can adjust its movement to re-center itself over the line.
- **Control Mechanism**: The sensor is connected to a microcontroller that processes the data from the sensors. Based on this input, the robot adjusts the motor speeds to maintain its path along the line. With five sensors, the system provides higher accuracy and allows smoother adjustments than configurations with fewer sensors.

This sensor setup is particularly useful for tasks such as autonomous navigation, path tracking, and educational robotics projects. The 5-channel configuration enhances the

robot's ability to accurately follow lines, including curves or more complex paths, by offering more precise feedback and control.

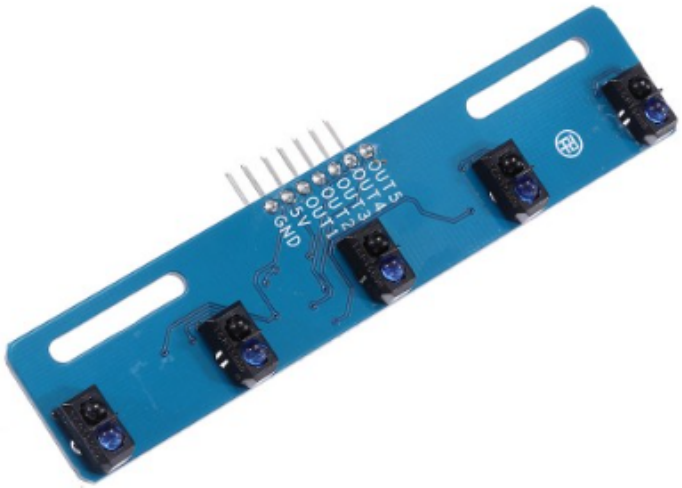


Figure 3.11: Line Follower 5 Channels Infrared Sensor Module

3.4 Battery

11.1 V Lipo Battery to supply all the motors of the robot, Lipo batteries are widely recognized for their high energy density, lightweight, and ability to deliver high currents, making them ideal for applications requiring large, consistent power.
Our Lipo battery is characterized by a voltage of 11.1V and a capacity of 5200 mAh. These features provide stable and durable power, which is responsible for managing and controlling the robot's motors. By using this battery, we ensured optimal performance of the power stage, allowing the robot to operate efficiently and meet power requirements during mobility operations.

Figure 3.12: LiPo Battery

3.5 Control Boards

3.5.1 Arduino MEGA 2560

Description

Arduino Mega 2560 is an exemplary development board dedicated for building extensive applications as compared to other maker boards by Arduino. The board accommodates the ATmega2560 micro-controller, which operates at a frequency of 16 MHz. The board contains 54 digital input/output pins, 16 analog inputs, 4 UARTs (hardware serial ports), a USB connection, a power jack, an ICSP header, and a reset button.

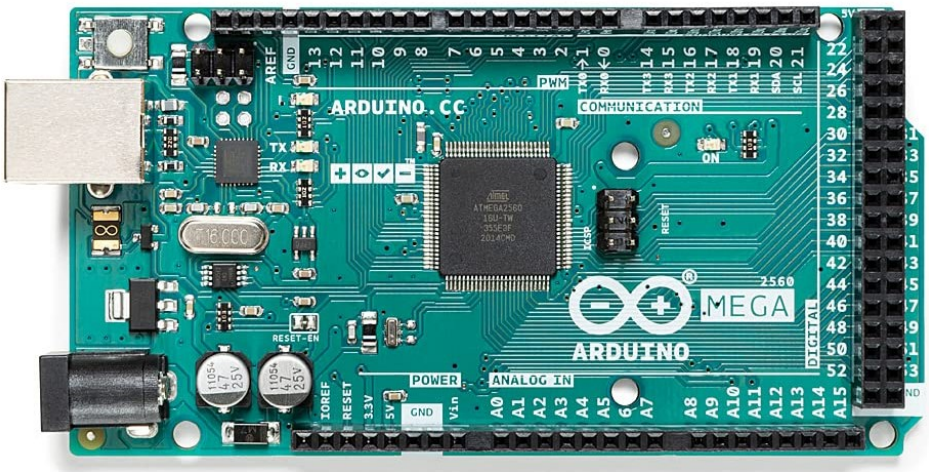


Figure 3.13: Arduino Mega 2560

Tech Specs

The Arduino Mega 2560 is a micro controller board based on the ATmega2560. Here are its technical specifications

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (including 15 PWM outputs)
Analog Input Pins	16
Flash Memory	256 KB
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
Dimensions	101.52mm x 53.3mm

Table 3.4: Arduino Mega 2560 Tech Specs

Connector Pin-Out

The Arduino Mega 2560 features a versatile pinout design that accommodates a wide range of applications and connections for various sensors, actuators, and devices. It includes 54 digital input/output (I/O) pins, numbered from 0 to 53, which can be utilized for both input and output operations. Notably, pins 0 and 1 are designated for serial communication, functioning as the transmit (TX) and receive (RX) pins, respectively. For analog input, the board provides 16 analog pins labeled A0 to A15, capable of reading varying voltage levels from 0 to 5V; additionally, A0 to A5 can serve as digital pins if needed.
The Mega 2560 supports Pulse Width Modulation (PWM) output on several digital pins, specifically 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 44, 45, and 46. For serial communication, it includes multiple UART interfaces, with dedicated RX and TX pins: 0 (RX0), 1 (TX0), 18 (RX1), 19 (TX1), 16 (RX2), 17 (TX2), 14 (RX3), and 15 (TX3). The board also features dedicated I2C pins, A4 (SDA) and A5 (SCL), for I2C communication, as well as SPI pins including 50 (MISO), 51 (MOSI), 52 (SCK), and 53 (SS) for SPI communication. Powering the Mega 2560 is made simple with a Vin pin that accepts a voltage input of 7-12V, alongside several ground (GND) pins, a regulated 5V output pin, and a 3.3V output pin (with a maximum current of 50 mA). The RESET pin allows for easy resetting of the Arduino board. The Arduino Mega is compatible with a variety of sensors, actuators, and displays thanks to its large input/output options and specific functionalities, such as PWM outputs, interrupt inputs, and communication pins, making it suitable for complex applications. This extensive pin configuration makes the Arduino Mega 2560 an excellent choice for projects that require numerous peripheral connections. For further details, one can refer to the official Arduino documentation.

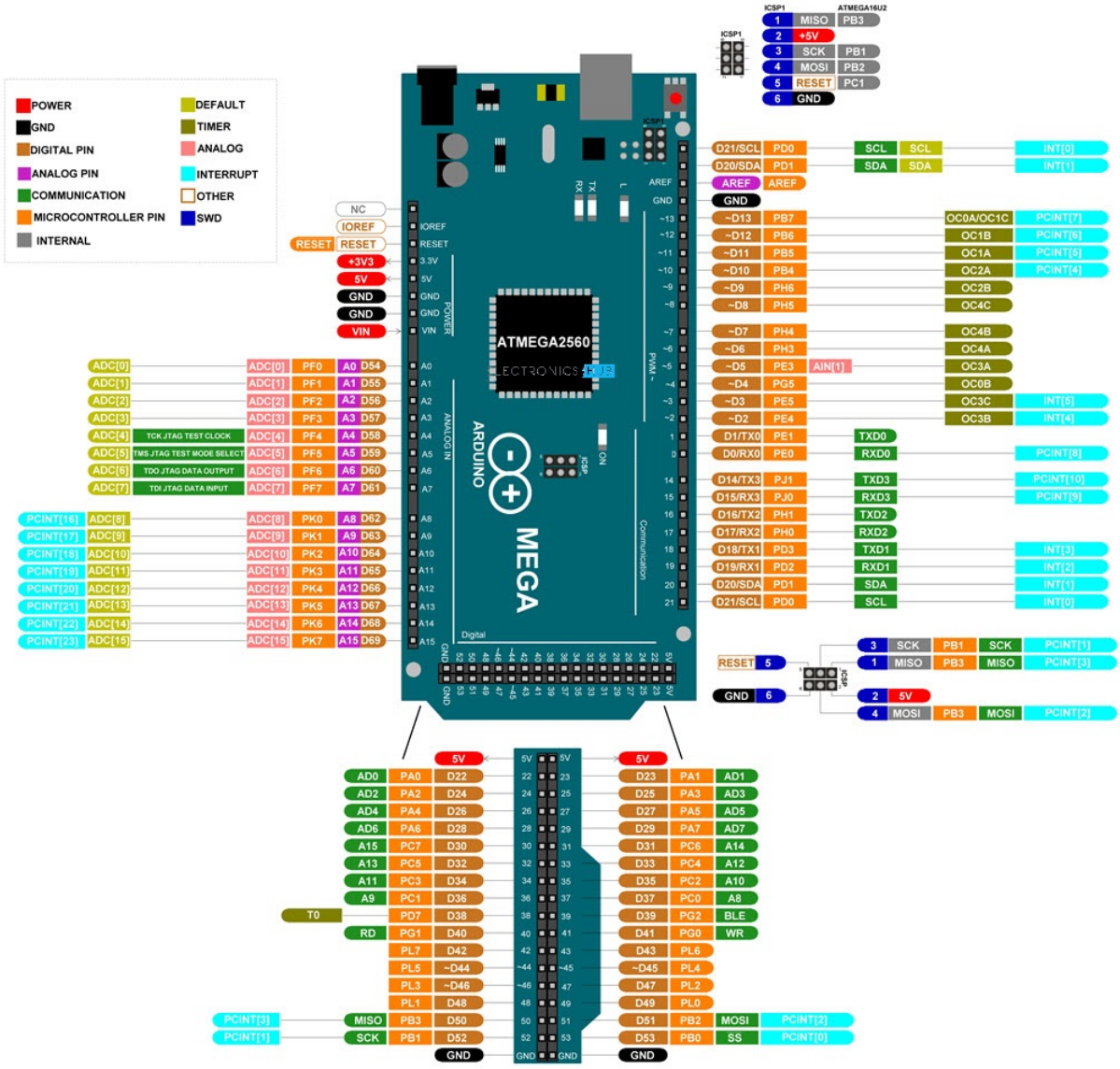


Figure 3.14: Arduino Mega 2560 Pin-Out

Application

The Arduino is commonly used due its large number of analog and digital inputs/outputs adding to that being compatible with the most of Arduino shields allowing it to be used in application as:

- Robotics: the high processing capacity that can handle extensive robotic applications and the ability to control different types of motors and sensor readings, making it suitable for the field of robotics.
- 3D Printing:Arduino Mega 2560 is capable to support the necessary algorithms for 3D printing and we mention the ability to change the code and customize it according to the requirements.

- Wi-Fi: the wireless connection between devices offers a variety of applications as the smart houses control using the Wifi shield.

3.5.2 Raspberry Pi 4 Model B

The Raspberry Pi 4 Model B is a compact single board computer known for its powerful performance offering a wide range of application and prototyping or even educational purposes in the fields of embedded systems or real time systems. This computer is equipped with a Linux operating system called Debian. It can be connected directly to a typical user interface using HDMI or a user interface accessible via SSH.

Raspberry Pi Characteristics

- Processor: Quad-core ARM Cortex-A72 (ARM v8) 64-bit SoC at 1.5GHz, in order to secure smooth and fast tasks running for general computing use or multimedia applications.
- Memory: Among the other versions we chose 8GB of LPDDR4-3200 SDRAM for high demands applications, including running multiple tasks simultaneously for example the image processing, the autonomous navigation and the VNC software at the same time or even using it as a server.
- Graphics: Broadcom Video Core VI GPU, supporting OpenGL ES 3.x, 4Kp60 hardware decode of HEVC video, dual-display output via two micro-HDMI ports supporting up to 4K resolution each.
- Connectivity:
  - Wireless: Dual-band 802.11ac Wi-Fi and Bluetooth 5.0 provide robust wireless communication.
  - Wired: The Gigabit Ethernet supplying high demands network tasks offering a fast wired connection to the network.
  - USB: Two USB 3.0 and two USB 2.0 ports offering fast data transfer.
  - GPIO: A 40-pin GPIO for the communication or the connection of sensors, motors, and other electronics.
- Storage: We can use an all purpose MicroSD card including the user files storage and the operating system files which can be a Debian, Ubuntu or others. we can use the USB ports for adding an external storage such as SSDs or flash drivers
- Power: The raspberry Pi gets supplied with energy via a UCB type C connector offering 5V and 3A for a continuous execution even under heavy tasks load.
- Display and Camera Interfaces: MIPI DSI for display and MIPI CSI for camera, supporting the Raspberry Pi camera and display accessories.

Applications

The Raspberry Pi 4 Model B is offering an ideal affordable computer which can be used for hobbies, education and professional uses for example:

- Learning programming and electronics.
- Building automated systems.
- Building a customized gaming console.
- Prototyping and IoT projects.
- Lightweight server hosting.

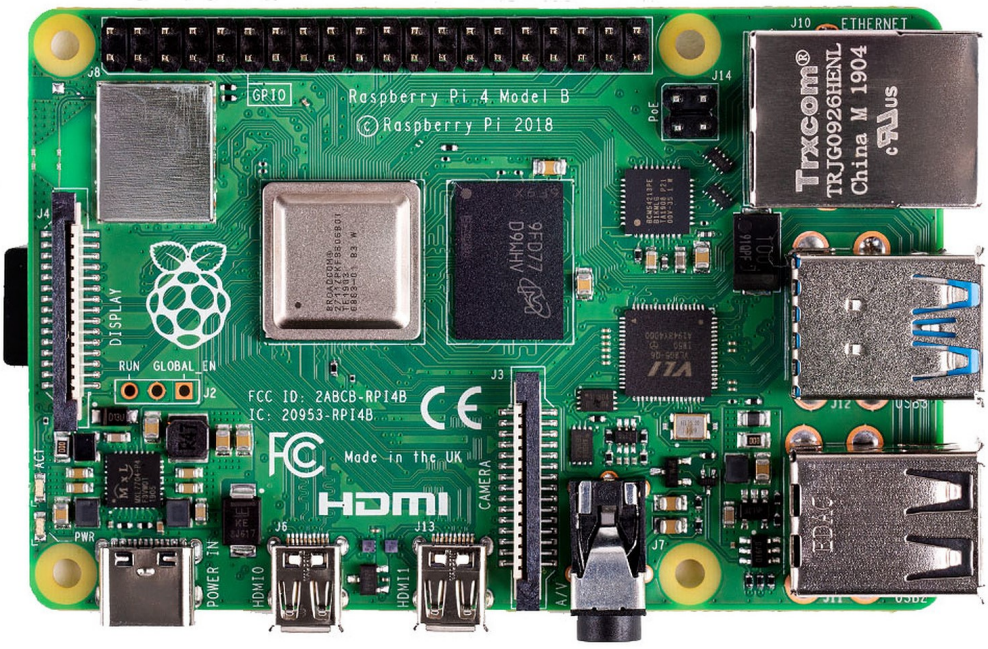


Figure 3.15: Raspberry Pi 4 Model B

Chapter 4

Building and Assembling a Mobile Manipulator Robot

For our project we ensured to use efficient components and motors to achieve a sophisticated functionality and robustness, we have combined the mobility offered by the mecanum wheeled robot with the Braccio TinkerKit robotic arm reach capabilities. This combination provides a wide range of applications for precise object manipulation and movement in diverse directions and ground natures.

4.1 Machines Utilized in the Project

We have used two main machines to create the mechanical part of the robot which are:

CNC Laser Cutting Machine

The CNC laser cutting machine is an automated equipment with a computer-controlled laser beam used for cutting materials like forex, plastic, wood and aluminum. The use of the machine needs the creation or importing a model or design into a Computer-Aided Design (CAD) software such as Solid Works. This design is then converted into a DXF file, the file is later sent to the machine which uses the information to control the laser beam so that it cuts material in exact precision according to the design.
In our case study, we applied this technology in creating the mobile robot chassis by cutting floors from black forex using the CNC laser cutting machine. The use of this technology enabled us to obtain a precise high quality cuts based on our design details and to reduce time.

3D Printer

The device used to make three-dimensional objects is a 3D printer by adding material in layers according to the model created from a digital design. The material resin or plastic

which is most of the time of the type PLA, Before sending the model to the printer you need to upload it on CAD software (for example Solid Works) and design the model according to the printer capacities like size and the maximum smoothness or robustness. The format of file should be an STL file type, then upload it into a 3D printing software, then once the piece file is uploaded to the 3D printer the loaded printing material will be heated and placed layer by layer.
In our case study, we have used the 3D printers during the project conception producing various parts for mobile robot manipulator like the sides cover, the mecanum wheels, the robotic arm base, links and gripper, the gimbal robot and the Raspberry Pi cover based on specific measurements.
The following Figure (4.1) and (4.2) shows an example of the used CNC and 3D printer machines respectively which are located in the FAB-LAB fabrication laboratory situated in Tlemcen Algeria.

Figure 4.1: CNC Laser Cutting Machine

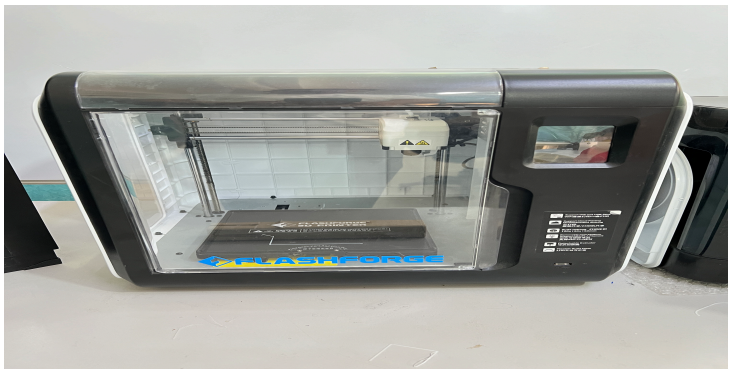


Figure 4.2: 3D Printer

4.2 Robotic Arm

4.2.1 First Robotic Arm Model

During the first days of implementation, we focused on creating a simple to implement and study 3D model, taking into consideration the dimensional limits. As a result of our research, we have chosen the 5 degrees of freedom robotic arm shown in the following figure

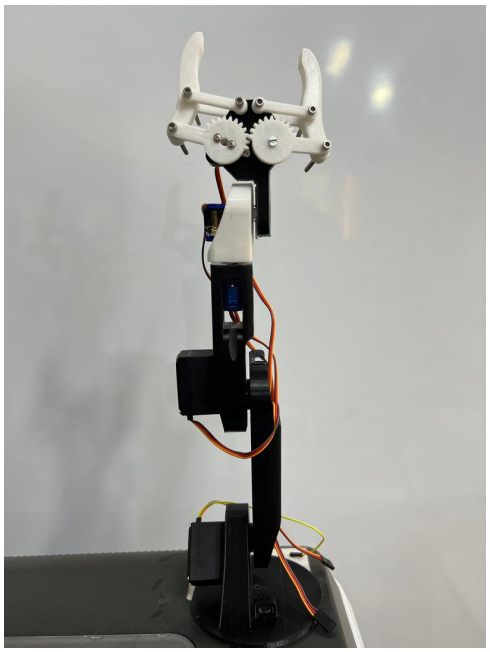


Figure 4.3: Design and Structure of the Previous Robotic Arm

Pros and Cons of the Model

From a part the last robotic arm has a number of advantages and we mention the most important of them including:

- The robotic arm is a well-known in the community of robotics which is going to help us in finding researches.
- The simplicity to print because of it lacks of sharp details and it reasonable dimensions and it has few pieces to implement.
- The robotic arm uses only servo motors MG995/MG996 and MG09 which are known for theirs availability and their light weight. From another part we should change the robotic arm for few reasons such as:
- The robotic arm has an unsuitable reach field which is going to reduce the manipulator capability and the mobile robot dimensions.

- We took in consideration the load applied on the servo motor shaft which is the only thing holding the arm from falling apart, and we mention the current consumption without avail for that cause.
- The robotic arm uses the MG09 in different joints which is not suitable because of it imprecision and it low torque and the capability of breaking down at any moment causing random movements capable to destroy the system parts.

4.2.2 Robotic Arm

We 3D printed a robotic arm model composed of 21 separate parts that can be assembled to offer smooth control of movements. Designed with precision and attention to detail, including hiding motors and wires, this robotic arm enables numerous applications to meet various needs.
The following Figure (4.4) and (4.5) shows the robotic arm different part and the final assembled arm respectively for clarification.

Figure 4.4: Parts of a Robotic Arm

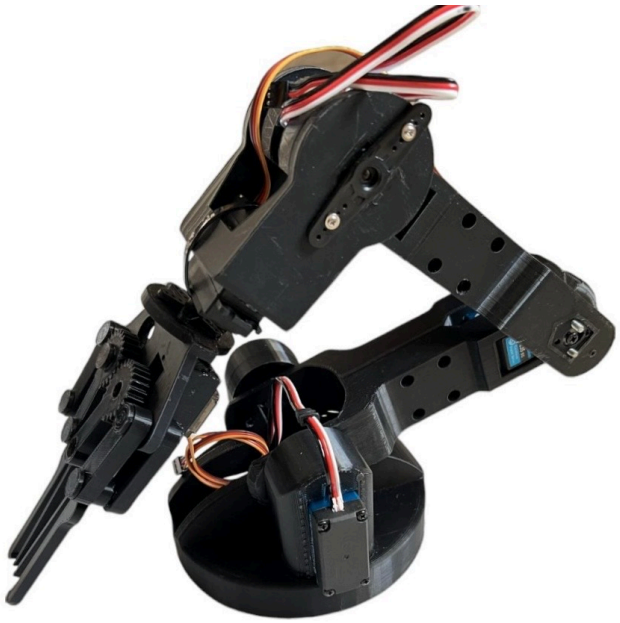


Figure 4.5: Final Assembly of the Robotic Arm

Pros and Cons of the Model

The distributed load on the arm and the MG995 and modification to hold the MG995 This model unlike the first one for such reasons and we mention their advantages and disadvantages and we start with:

- This robotic arm as a considerable dimension difference offering a wider range of reach ability and more strength joints and links.
- The manipulator has a distribution load system for the joints efficient which causes the stability of movements around the axes and in this way, it applied less pressure on the motor shaft leading to energy reduction and durability of the components.
- We observe the aesthetic and the appearance difference of the braccio robotic arm
- The model is composed of separated part offering space to hide motors and wires.
- We mention the mass difference which counts as a disadvantage obliging us to use a high torque servo motor and occupied metallic gear shaft.

4.3 Mobile Robot

As the robotic arm also the mobile robot have seen some changes during the development from the look and the functionality for example at first it was made of wood equipped with 4 holes designed to connect the stepper motors shaft the 3d printed mecanum wheels the ones shown in the Figure (4.6) for the distance crossing as it shown in the Figure (4.7), but we have changed the stepper motors to the JGA25 370 motors which are significantly smaller and lighter which offers the option to put them outside the car and benefit from the new free space.

Figure 4.6: Detailed View of 3D Printed Mecanum Wheels

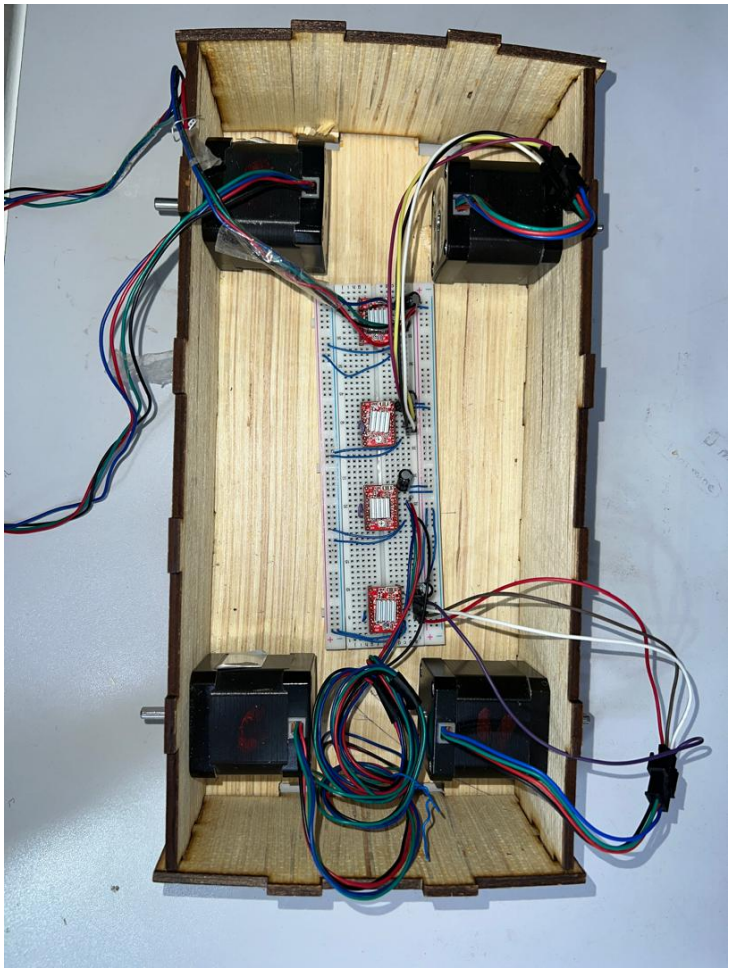
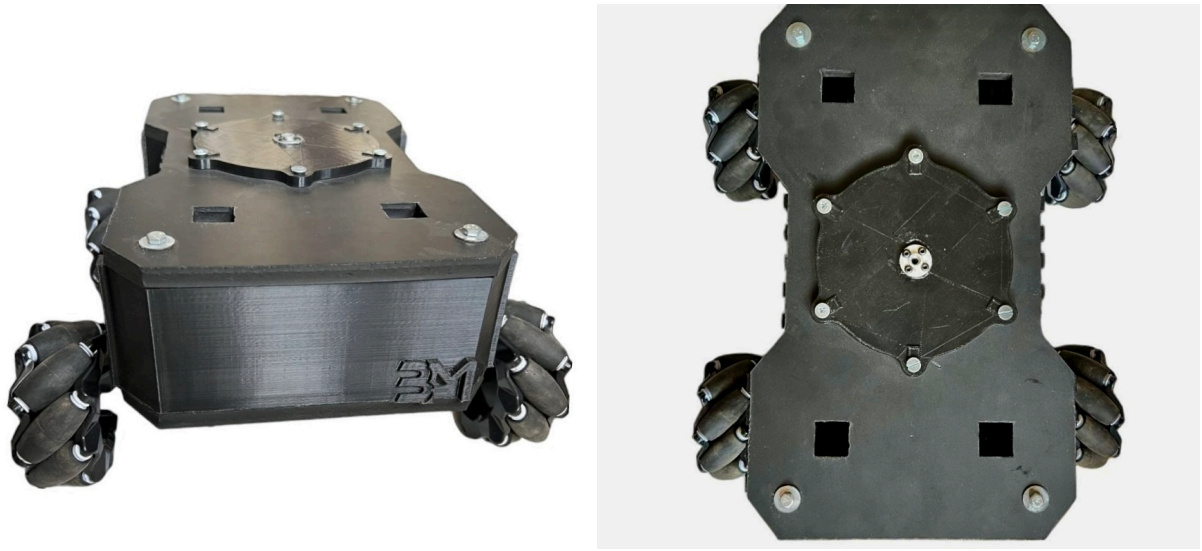


Figure 4.7: Components of the Previous Mobile Robot, Including Motors and Drivers

The mobile robot is the heaviest and most robust part of the robot body, as it should stabilize movements and prevent shock damage. The robot body can be created using 3D printing and a CNC laser cutting machine and is made of PLA (Polylactic acid).
The 3D printed sides cover is screwed with the forex platform which represents the base and the top of the chassis. We ensured to calculate and make all mechanical or electrical needs such as wires inputs to the Arduino and the other components inside, the battery charger, the Arduino USB type B cable, the battery state switch (consumption / charge / OFF), low current 12V adapter for tests, the motor supports and wires placement, The robotic arm is placed on top of the robot and fixed with 7 metallic screws with and intermediate part connected directly to the robot base servo motor. All what is mentioned before in shown in the Figure (4.8) presenting the mobile robot.



(a) Front View of the Mobile Robot (b) Top View of the Mobile Robot

Figure 4.8: Perspectives of the Mobile Robot

4.4 Architecture

The hardware architecture proposed plays the critical role in the functionality of the robot to achieve a task that is desired including autonomous navigation and the objects pick and place. It consists of components and interconnections between electronic circuits ensuring the desired goal of this combination.
The architecture is composed of high-level control and low-level control. The high-level segment deals with managing the navigation, guidance algorithms of the robotic platform, the camera vision object detection, deciding the next desired positions and the trajectory planning optimization in parallel. while the low-level one ensures execution of orders sent from high level by piloting four motors of the robotic platform and the 7 motors of the robotic arm using PWM signals.

4.4.1 High-Level Control

The high-level segment consists of The Raspberry Pi4 model B calculator which communicates with the main station (PC) via a wireless connection (WIFI) or an HMI (human machine interface) to insert data and orders.
The Raspberry Pi allows remote control from another computer to set the desired working area, then this calculator will optimize the trajectory by the A* method and when it is arrived the camera start scanning the area and share the real time video with operator, after detecting the object in our case it is represented as an QR code then thank to the image processing it can define the object coordinate and send it to the Arduino to insert it into the inverse model of the robotic arm the get the required angles, each time the a task is done by the low level controller it will notify the Raspberry Pi to secure the task

sequence.
The Raspberry is powered by a power bank which ensures a secure and a stable 5V power supply for the Raspberry rather than a Lipo battery of 11.1V. A power bank is a portable energy storage device used to charge or power electronic devices. It contains an internal rechargeable battery and has a micro-USB port for connecting various devices. The power bank we selected has a capacity of 20,000 mAh and provides an output voltage of 5V and equipped with a digital battery display which allows the operator to know the remaining charge level.

4.4.2 Low-Level Control

for the low-level segment, it is composed of an Arduino Mega microcontroller which controls the four DC motors and 7 servo motors. To set up an odometry module and guarantee speed control of the motors, we used four JGA25-370 type motors equipped with encoders. The 11.1V battery (lithium polymers with a capacity of 5200 mAh) powers the DC motors and the servo motors using a boost step backward forwards converter to achieve the stable 5V supply power. This dual power configuration provides protection for the on-board system by isolating the low-power and high- or medium-power parts, while ensuring better overall system autonomy. The main role dedicated to this segment is to ensure low-level control of the robot including the speed control and the achieve the desired position for the platform and the arm gripper.
The Low level control has too many components which makes the process of wiring without damaging the parts a complex issue and for that we used the diagram shown in the Figure (4.9).

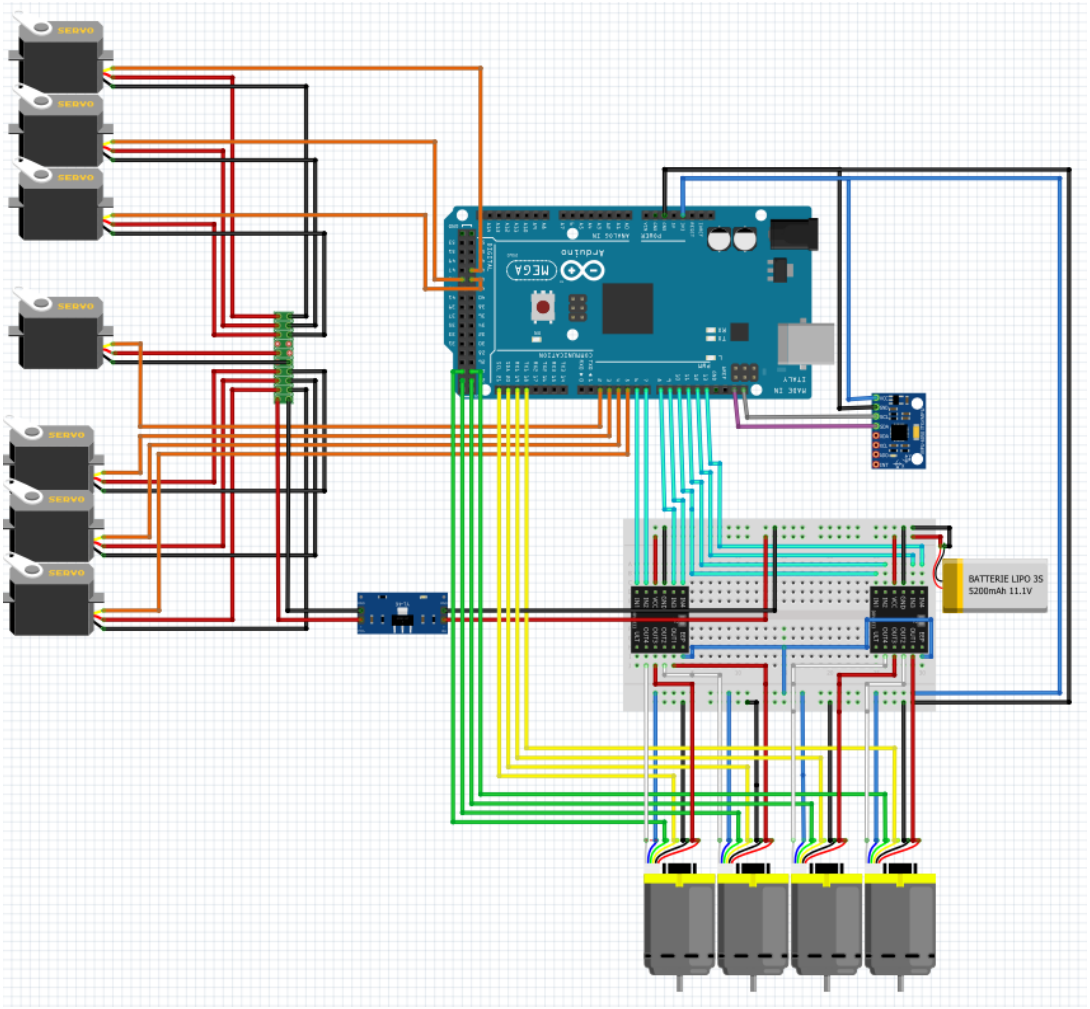


Figure 4.9: Low-Level Control Wiring

4.4.3 Communication Protocols in Mobile Robot Manipulators

The mobile robot manipulator consists of multiple controllers which need communication between them to ensure data transmission and receive. The raspberry pi and the Arduino communication in embedded systems projects is used to combine the powerful multi core processor of the raspberry pi and the multi-inputs outputs capacity of the Arduino. This bidirectional interaction and exchange of data and commands is insured with the serial communication using the USB type-B cable. To ensure a wireless and long distances control and visualization we used the WIFI connection between the user computer (master) and the raspberry pi (slave) as a communication method. This was possible using the ssh and the VNC application which offers a real time view and access to the raspberry pi to control and modify values [11].
In the next Figure (4.10) we can see the different robot's components and their role in control mentioning the used communication types and the content of each signal.

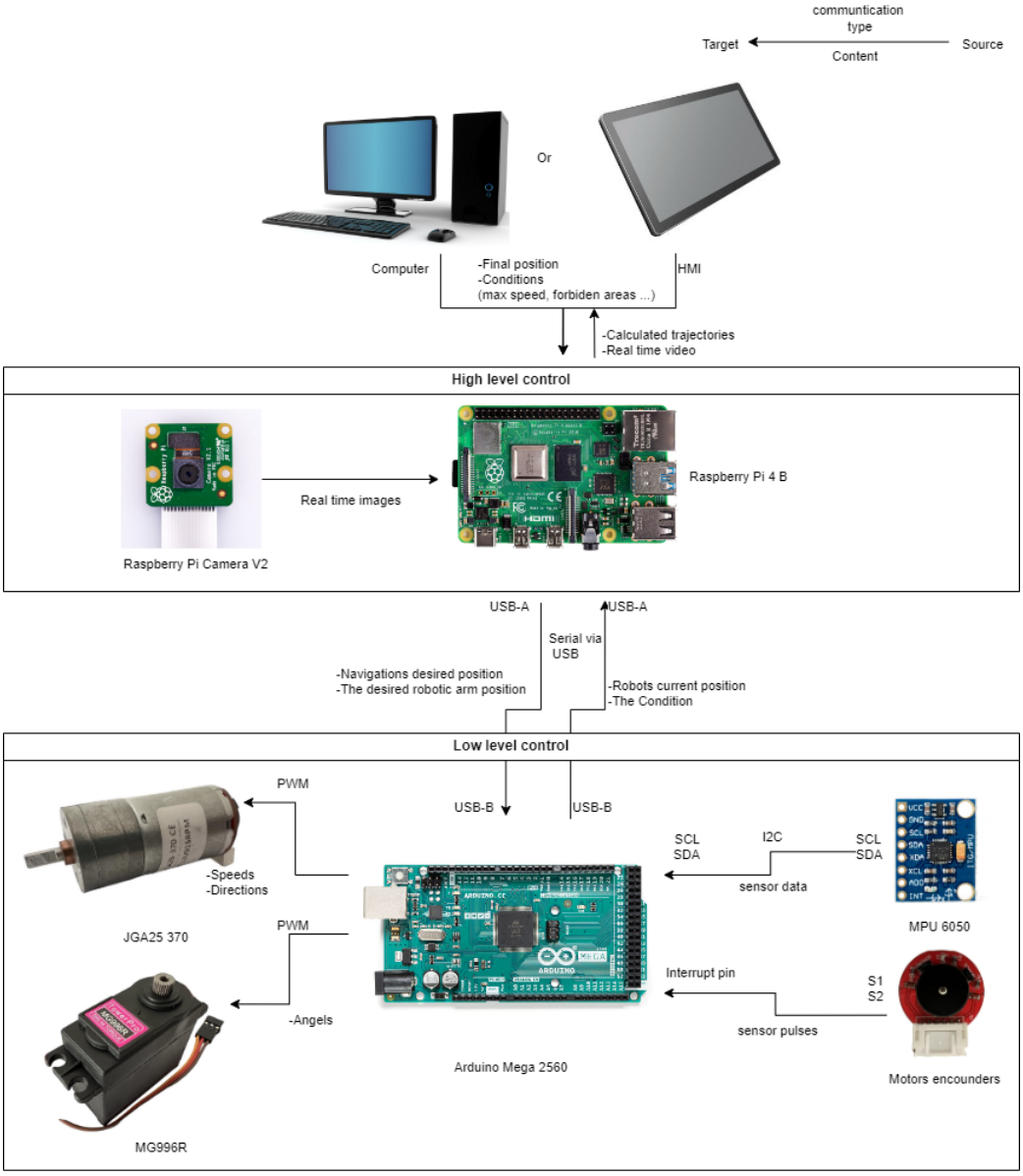


Figure 4.10: Control Levels and Communication Architecture of the Robot

4.5 Assembly of Mechanical and Electronic Components in Mobile Robot Manipulators

After the creation of the mechanical part and the electronic connections we had to assemble the two parts to create the mobile robot manipulator. This robot is constructed of two floors and each one represents a level of command and segmentation. In the first floor we find the JGA25 370 motors, the power stage, the electronic and control components including the power Bank, the Lipo battery, the 12v to 5v boost step-up step-down converter and, the DRV8833 motors drivers and the Arduino mega the low-level segment controller, all these components are soldered in one permanent circuit to ensure the stability and safety and avoid the connections problems. The motors and the circuit are

attached with bolts to the base floor.
On the second floor theres three main structures the robotic arms with the servo motors and their wires, the two covers to protect and hide the raspberry bi 4B which is the high-level segment of command and its USB serial cable and finally the last structure is the camera gimbal offering the 270° angle of vision.
The following Figure (4.11) and (4.12) shows the early mentioned parts assembled and the electronic circuit respectively.

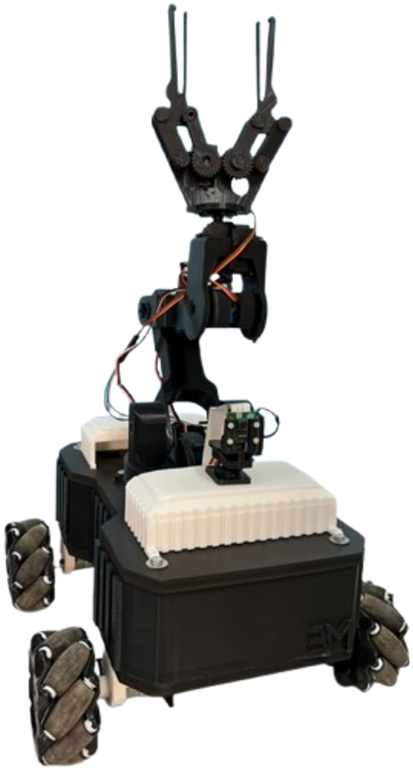


Figure 4.11: Assembled Mobile Robot Manipulator

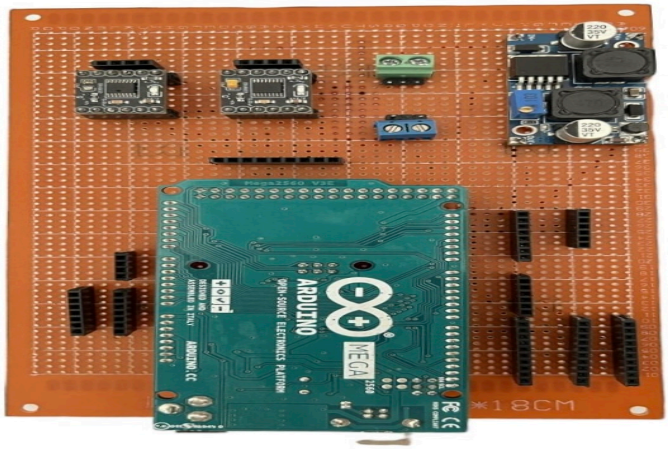


Figure 4.12: Electronic Circuit Design and Components

Chapter 5

Mobile Robot Manipulator Systems: Control and Integration

Choosing the best components and mechanical parts is not enough in the robot development; we need to add another factor, which is robot control.
Controlling a mobile robot equipped with an arm presents a complex and dynamic challenge in robotics, blending locomotion and manipulation. This system combines two essential functions: the movement of the mobile base and the precision of the robotic arm. The mobile robots base allows for navigation in various environments, while the robotic arm performs tasks such as picking, placing, or interacting with objects. Coordinating these two systems requires advanced control algorithms, sensor integration, and motion planning to ensure smooth and efficient operation. By leveraging feedback from sensors such as cameras, LiDAR, or encoders, the robot can perceive its environment, plan trajectories, and execute tasks with accuracy.

5.1 Control Process in Robotics

The control process involves continuously adjusting and regulating the robot's parameters to achieve a desired outcome, such as precise movement or positioning. Controlling and automating robotic systems is essential, especially for low-level commands such as speed, position, direction, stability, and arm movement. The complexity of control depends on the variation of the system's parameters, but there are key benefits to this approach:

- **Energy Efficiency**: By dynamically adjusting energy input according to the manipulator's load requirements, the controller minimizes power consumption, avoiding waste and enhancing overall energy efficiency.
- **Error Elimination**: Automated control systems in robotic arms ensure continuous monitoring and fine-tuning, reducing the likelihood of human error and maintaining precision in operation.

- **Quality Assurance**: Repeated tasks, such as gripping and positioning, benefit from automation, ensuring that accuracy and repeatability are maintained throughout the manipulators movements.
- **Improved Safety**: Safety mechanisms like emergency shutdowns and collision prevention systems are integral to automated control, reducing the risk of damage to the robot or its surroundings.

Inputs

The inputs to the manipulator's control system consist of data from sensors (position, velocity, torque, etc.), energy supplies, and commands that dictate the desired movements. While the robot follows programmed instructions, it must also adapt to minor variations in sensor feedback, mechanical components, or load conditions. These variations should be within tolerance limits to maintain the robot's proper functioning.

Uncontrolled Variables

Uncontrolled variables in robotic manipulation include external environmental factors, such as ambient temperature, humidity, or vibrations, which may influence the robot's performance. These factors cannot be directly controlled by the robot's system. Additionally, variations in the robot's joints or actuators due to wear and tear, or differences in operational load, can introduce variability into the system.

Controlled Variables

Controlled variables include the speed, position, direction, and stability of the manipulator's joints and end-effector. Sensors monitor these parameters and send feedback to the robot's control system, which uses algorithms (e.g., PID controllers) to adjust motor torque, arm position, or joint angles to achieve the desired performance. Advanced control systems may also incorporate artificial intelligence to optimize control strategies in real time.

5.2 Controlling a Mobile Robot with an Arm

Controlling a mobile robot with an arm involves complex coordination between the bases motion and the arms manipulation tasks. The control problem is typically split into two parts: controlling the mobile base (which often has wheels or tracks) and controlling the robotic arm (which consists of multiple joints or degrees of freedom).

Mobile Base Control

The mobile base requires navigation and path-planning techniques, usually handled through feedback control algorithms. These algorithms use data from sensors like encoders, inertial measurement units (IMUs), or GPS to compute the desired velocity and position while avoiding obstacles and maintaining stability. Common controllers include PID (Proportional-Integral-Derivative) controllers or more advanced methods like Model Predictive Control (MPC). The control architecture must handle external disturbances (like uneven terrain) and maintain precise positioning for the arm to interact with objects.

Arm Control

The robotic arm, which is attached to the mobile platform, has multiple joints that need to be controlled in a coordinated manner. Each joint is typically driven by individual motors, and controlling these requires precise trajectory planning and kinematic control. Inverse kinematics (IK) is often used to compute the required joint angles to achieve a specific end-effector position. For dynamic tasks, dynamic control is applied, considering forces, torques, and inertia to maintain stability, especially when the arm interacts with the environment.

Unified Control

The key challenge is unifying the control of both systems so that the robot can move and manipulate simultaneously. This requires integrating data from various sensors (cameras, LiDAR, etc.) to localize the robot, understand the surroundings, and plan movements. Advanced control schemes like task-space control or operational space control are often used to manage the robots interactions with the environment by decoupling the base and arm controls, allowing them to work harmoniously.
One major challenge in controlling such systems is compensating for the dynamic coupling between the mobile base and the arm. Movement of the mobile base can affect the stability and positioning of the arm, and vice versa. Therefore, the control algorithms need to predict and counterbalance these effects in real-time.

5.3 Control Architecture of Mobile Robotic Manipulators

For our project involving a mobile robot manipulator, we have chosen a hybrid control architecture that balances strategic planning and real-time adaptability across high-level and low-level control systems [13].

High-Level Control

- **Trajectory Planning and Target Definition:** The high-level control is responsible for defining the path the manipulator will take, determining joint configurations, and setting the final target position. This involves solving complex kinematics and optimizing the movement sequence to ensure efficiency and precision.

Low-Level Control

- **Application and Motion Control:** The low-level control handles the execution of the trajectory, translating the high-level commands into precise motor actions and sensor integration to achieve the desired motion.
- **Improvisation and Disturbance Rejection:** The low-level control can independently handle real-time feedback, such as resisting external perturbations (e.g., unplanned forces or disturbances). The system can adjust without needing to wait for high-level intervention, improving the manipulator's responsiveness and stability.

Advantages of This Approach

- **Real-Time Adaptation:** The low-level controllers ability to react to unexpected disturbances (like external forces) ensures robustness and precision. This is particularly useful in manipulative tasks that require consistent interaction with unpredictable environments.
- **Efficiency:** The high-level controller can focus on broader goals, leaving real-time, detailed adjustments to the low-level controller. This reduces the computational burden on the high-level system, allowing for more efficient control.
- **Improvisation:** Low-level control can manage perturbations and minor corrections without needing continuous input from the high-level control, improving the overall autonomy of the system.

5.4 Low-Level Control

We will focus on the low-level control systems of robotic manipulators due to their significant impact on the system's performance and behavior. Low-level control encompasses the regulation of essential parameters such as speed, position, direction, stability, and torque. These aspects are crucial because they directly influence the robot's precision and adaptability in real-time. The variation in these control parameters is substantial, and fine-tuning them is vital to achieving optimal performance. By concentrating on low-level control, we can address the dynamic challenges and fluctuations that arise during

operation, ensuring that the manipulator responds accurately and efficiently to varying conditions and tasks. We can handle the changes and challenges that come up during operation, making sure the robot responds accurately and efficiently to different conditions and tasks. In the following table we can observe the different tasks ,possible variable regulations and systems that could be done by the robot's manipulator low-level segment [14].

System	Description	Purpose	Regulation Components
Speed Control System	Regulates motor speed controlling joints and end-effector.	Ensures smooth and controlled motion, preventing unsafe movements.	Encoders, tachometers, motor drivers.
Position Control System	Controls the precise positioning of joints and end-effector.	Ensures accurate positioning, critical for tasks like picking or placing.	Encoders, resolvers, feedback loops (e.g., PID controllers).
Direction Control System	Manages orientation and trajectory of arm and end-effector.	Regulates movement direction for precise path planning and alignment.	IMUs, gyroscopes, control algorithms.
Stability Control System	Maintains stability of the arm, preventing oscillations or overshooting.	Ensures steady operation, especially with varying loads or quick movements.	Force/torque sensors, IMUs, adaptive/robust control algorithms.
Torque Control System	Regulates torque applied by motors to each joint or axis.	Adjusts force for moving/holding the arm, ensuring energy efficiency and safety.	Torque sensors, current sensors, motor controllers.
End-Effector Control System	Regulates actions of the end-effector (e.g., gripper, tool).	Controls gripping force or tool movements for specific tasks.	Force sensors, pressure sensors, motor drivers for actuators.
Arm Trajectory Control System	Manages path and motion planning for arm movement.	Optimizes path for efficiency, avoiding obstacles and minimizing time.	Path planning algorithms, kinematic/dynamic models, position feedback sensors.
Load Compensation System	Regulates manipulator response when handling different loads.	Adjusts motor torque, speed, and stability for load variations.	Load sensors, force/torque feedback, dynamic adjustment algorithms.
Temperature Control System	Regulates temperature of motors and electronics.	Prevents overheating during prolonged or intensive operations.	Temperature sensors, cooling fans, thermal management systems.
Force Feedback System	Monitors and regulates force exerted by the end-effector.	Ensures correct amount of force without damaging objects.	Force sensors, feedback loops integrated into control algorithms.
Power/Energy Management System	Regulates power supplied to actuators and control systems.	Optimizes energy use, avoiding overloading or under-powering motors.	Voltage/current sensors, power distribution units, energy optimization algorithms.
Collision Avoidance System	Regulates robot motion to avoid collisions with obstacles or humans.	Enhances safety by stopping or rerouting movement to prevent collisions.	Proximity sensors, vision systems, real-time control software.
Vibration Dampening System	Regulates vibrations caused by manipulator movements.	Minimizes oscillations or vibrations affecting precision or mechanical wear.	Vibration sensors, damping control mechanisms.
Grip Force Control System	Controls gripping force of the robots gripper.	Prevents damage to objects while ensuring a secure grip, adjusting for material properties.	Pressure sensors, tactile sensors, motor controllers.
Communication and Data Exchange System	Manages data flow between sensors, actuators, and control	Ensures timely data exchange and control commands for	Data buses, communication protocols (e.g., CAN, EtherCAT),

In our system architecture, the low-level control tasks includes motor speed control ensures the smooth operation of actuators, maintaining the desired velocity even under varying load conditions, The regulation of the robot's speed and position is crucial for accurate navigation and task execution, while yaw angle control allows for precise adjustments in orientation, essential for tasks requiring directional changes. Additionally servo motor control, enables fine regulation of speed and position, ensuring that actuators perform tasks with high accuracy. As the low-level controller, it is responsible for real-time execution of these critical functions, ensuring smooth operation and precise movements by utilizing PID controllers for these key control elements.

5.5 Proportional Integral Derivative (PID) Controller Overview

The Proportional-Integral-Derivative controller has a rich history dating back to the early 20th century, when it was developed for industrial process control, particularly in the field of automatic steering for ships by Elmer Sperry in 1911. Its wide adoption is due to its simplicity and effectiveness, as it requires no complex mathematical model of the motor or robot it controls, making it particularly advantageous for systems where precise modeling is difficult or unnecessary [15]. A key reason for choosing PID in modern robotics and motor control is its ease of implementation and tuning, allowing it to efficiently handle a variety of tasks, from position control to speed and torque adjustments. Moreover, PID controllers are highly adaptable to changing conditions, capable of correcting errors in real-time without needing intricate system knowledge, making them ideal for scenarios where motor dynamics are unknown or where the system experiences frequent disturbances. These benefits, along with its well documented history and proven reliability, making the PID a solution in control systems for robotics and beyond.

5.5.1 PID Controller Theory

The PID controller is a widely used feedback control mechanism designed to minimize the error between a desired set-point and a measured process variable. It computes a control output based on three distinct components which are the proportional, integral, derivative gains [16].
The overall control output $u(t)$ is given by:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\,d\tau + K_d \frac{de(t)}{dt} \tag{5.1}$$

$u(t)$ is the control output at time t,

$e(t)$ is the error at time t (the difference between the set-point and the process variable),

K_p is the proportional gain,

K_i is the integral gain,

K_d is the derivative gain.

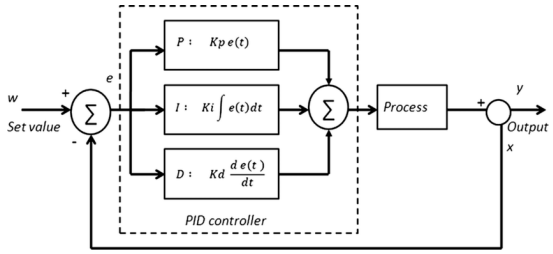


Figure 5.1: Diagram of the PID Controller Implementation

Proportional (P) Component

$$P(t) = K_p e(t) \tag{5.2}$$

A proportional controller responds to the current error $e(t)$ by applying a corrective action that is directly proportional to the error, with the proportional gain K_p determining the magnitude of this response. A higher K_p results in a stronger correction, which leads to faster error reduction and a quicker system response. However, excessively high values of K_p can cause the system to overshoot the target, potentially leading to oscillations or instability. A balanced K_p is crucial for achieving a stable, responsive system.

Integral (I) Component

$$I(t) = K_i \int_0^t e(\tau)\, d\tau \tag{5.3}$$

An integral controller addresses the accumulation of past errors by integrating the error over time, producing a corrective action that helps to eliminate steady-state errors. The integral gain K_i determines how strongly the controller responds to the accumulated error. While higher K_i values can effectively reduce steady-state error, they may also cause increased oscillations and a slower system response, as the system may overcompensate for errors that have already been corrected. Therefore, careful tuning of K_i is essential for maintaining stability and avoiding excessive oscillations.

Derivative (D) Component

$$D(t) = K_d \frac{de(t)}{dt} \tag{5.4}$$

A derivative controller predicts future errors by considering the rate of change of the error over time. The derivative gain K_d provides a damping effect, helping to reduce overshoot and improve system stability by slowing down the response as the error decreases. However, since the derivative action is highly sensitive to noise in the error signal, even small fluctuations can result in erratic control actions, making the system unstable. Proper tuning of K_d is necessary to balance the benefits of damping without introducing instability from noise sensitivity [17].

5.5.2 PID Controller Tuning

Tuning a PID controller involves adjusting the three parameters proportional gain K_p, integral gain K_i, and derivative gain K_d to achieve the desired control performance. Proper tuning ensures that the system responds accurately and stably to changes in set-point and disturbances. Here are common methods and points for tuning:

Manual Tuning

Manual tuning is a fundamental approach for setting the parameters of a Proportional-Integral-Derivative (PID) controller to achieve desired system performance. It is a method that relies on the control engineer's experience and intuition to adjust the controller parameters. This approach is often used when automated tuning methods are not available or practical. The Procedure of manual truning is:

1. Set K_i and K_d to zero. Gradually increase K_p until the system begins to oscillate.
2. Adjust K_i to eliminate steady-state error while minimizing oscillations.
3. Tune K_d to improve stability and reduce overshoot.

One of the main advantages of this tuning method is its simplicity and intuitive nature, as it does not require any specialized tools, making it accessible for manual adjustment of the controller's gains. However, a significant drawback is that the process can be time-consuming, as fine-tuning the gains often involves a trial-and-error approach, which may not result in optimal performance. Achieving the best possible balance between response speed, stability, and accuracy can be challenging without more sophisticated methods or automated tools.

Ziegler-Nichols Method

The Nichols-Ziegler method is a classical technique used in control systems engineering for designing and tuning controllers. Its often applied to the design of Proportional-Derivative-Integral (PID) controllers. This method involves using Nichols charts, which are used to visualize the frequency response of a system and to determine the appropriate controller parameters [18].

Software-Based Tuning

Software-based tuning utilizes automated tools that leverage algorithms to set PID parameters by analyzing system performance data. This method provides the benefit of enhanced precision and speed, allowing for real-time adjustments based on continuous feedback. However, it also has the drawback of necessitating access to specialized software and tools, which may not always be accessible [17].

Model-Based Tuning

Model-based tuning involves creating a mathematical model of the system and applying optimization techniques to determine the optimal PID parameters. This approach is advantageous because it allows for highly optimized control, especially for complex systems. However, it also has some drawbacks, including the need for detailed system modeling and significant computational resources, which can be demanding and time-consuming [18].

5.5.3 Limitations of PID Controllers

Noise Sensitivity

One issue with the derivative term (Kd) in PID control is that it can amplify noise in the error signal, which may result in erratic control actions. To mitigate this problem, filtering techniques can be applied, or a derivative filter can be used to smooth out the noise and reduce its impact on the control performance [18].

Integral Windup

Integral windup happens when the integral term in a PID controller keeps accumulating error during saturation, causing slow responses and prolonged oscillations. To fix this, you can use anti-windup techniques like limiting the integral term or adjusting it based on system conditions to keep control smooth and stable [18]

Over-Sensitivity to Tuning

Over-sensitivity to tuning occurs when PID parameters react too strongly to changes, making it hard to find the right balance between responsiveness and stability. To address this, robust tuning methods can be used to find more stable settings, and adaptive control strategies can be considered to adjust the parameters dynamically based on system performance.

Complexity in Nonlinear Systems

PID controllers may struggle in systems with significant nonlinearity or varying dynamics. To address this, you can combine PID with other control strategies or use advanced methods such as adaptive or nonlinear control techniques to better handle complex behaviors and improve performance.

Limited Performance in Systems with Large Delays

PID controllers can have difficulty managing systems with large time delays or lag. To improve performance in such cases, techniques like the Smith Predictor or Model Predictive Control (MPC) can be used, as they are designed to better handle and compensate for delays.

Lack of Optimality

PID controllers may not always provide optimal performance or stability, particularly in complex systems. To address this, you can explore more sophisticated control approaches or combine PID with other techniques to enhance overall control effectiveness.

5.6 Motor Control Strategies

To achieve the desired translation and rotation speeds for the robot we need a precise speed control of its four DC motors. This involves regulating each motor's speed to maintain constant rotation, even in the presence of external disturbances, minimizing internal errors that could compromise the accuracy of the localization system. In the following sections, we will first explain how we successfully controlled the speed of a single motor using a PID controller, justify the chosen parameters, and then move on to implementing speed control for all four motors.
Effective robot motor speed control involves understanding several core principles: Pulse Width Modulation (PWM) for regulating speed, encoders for measuring and calculating motor speed, and interruptions for real-time adjustments. Additionally, it encompasses the methods for changing both speed and direction to ensure precise and responsive control of robotic systems.

5.6.1 Principle concepts

Encoder and Feedback

The DC motors by their own can't be feedback controlled but with adding and encoder we can observe the angular position of the rotor and calculate its speed and use it in a regulation loop to achieve the best performance possible.

The Hall effect encoder is a type of position sensor that uses the Hall effect principle to detect changes in magnetic fields. It typically consists of a Hall sensor and a rotating magnet, where the sensor measures the magnetic field's variations as the magnet moves. These changes are then converted into electrical signals to determine the position, speed, or direction of a rotating object. Hall effect encoders are commonly used in motor control, robotics, and industrial automation due to their reliability and precision in harsh environments [6].
The encouder pulses are detected by the Arduino as an interruption. The interrupt is an event that changes the order in which the processor carries out instructions. It can either be planned (intentionally triggered by the active program) or unplanned (resulting from an event that may or may not be connected to the current program) [19].

Position Measurement

To measure position using the haul effect encoder we need a function beside our loop function, This function consists of a variable called an increment that we will add to or subtract of the previous position variable each depending on the direction of the movement, we have built an interruption function that will be activated each time our mictrocontroller detects a rising signal in a specified interrupt input pin, this last operation will pause the principal loop function and enters the interruption function which will verify the other encoder pin status (B), for example if the B pin is taking the value 0 (LOW), we can conclude that the direction of motion in anti clockwise in this case we add the increment to the previous position but if its the opposite then the motion is anti-clockwise and we need to subtract the increment.
To ensure that the position variable is stored and both the loop and interrupt functions can use it we work with the volatile qualifier, which prevents the compiler from performing optimizations on the variables,and this optimization can potentially misreading the position.The ATOMIC BLOCK macro is needed to prevents the interrupt from changing part of the position variable while it is being read.

Speed Calculation

The speed calculation is based on a mathematical equation consist of defining a time variable in this case we call it current Time using the function micros() then calculate w the difference between the previous time and the current one then divide the result by a large numbre to minimize the deltaT then divide the increments of position (the variation in position between the previous time and the current time) by the delta which is represents the difference between the previous position and the current one and how the velocity change during the deltaT period.
currentTime = microseconds()
DeltaT = (currentTime - previousTime)/10^6
velocity = increments / DeltaT
previousTime = currentTime

Directional Movement Control

In the simplest case of driving one DC motor, we can change direction using a HIGH LOW (1/0) logic where the HIGH state represents applying 5V to the IN1 pin and the LOW state represents connecting the Ground to the IN2 pin and changing the pins means changing the direction as it shows in the next table

Input A	Input B	Motor State
Low (0)	Low (0)	Motor OFF
High (1)	Low (0)	Forward
Low (0)	High (1)	Backward
High (1)	High (1)	Motor OFF

Table 5.2: Motor Direction Control

Speed Adjustment Techniques

We can change the speed by changing the Pulse Width Modulation (PWM) connected with the high IN pin offering a full control on the output voltage in a range of (0 to 100) of the VCC value which is usually 12V and in our case it can rotate the DC motors in full speed and in any desired speed.

Pulse Width Modulation

PWM signals in The case of the robot manipulator all the motors positions and speed control commands are analog values so they transferred from the micro controller to the motor in a Pulse Width Modulation signal format.
Since most micro controllers are equipped with a digital signal generator (1 or 0) only so it cant generate analog signals which represents the values between the 1 and 0, this issue exists because the high prices of the Digital-to-analog converters and their considerable area. As a cost-efficient solution we use the PWM signals.
PWM or Pulse Width Modulation is a method used to generate an analog signal using a digital one, allowing us to control motors, valves and more, by controlling the delivered current by switching the ON and OFF power state rapidly. The average power raises if we keep the on state longer than the off state and the opposite. In this way we can modify a digital signal to make a variety of average power which means a variety of values which can be translated to speeds or positions [16].
The most important parameter of the PWM signal is the duty cycle which represents the time we should keep the digital signal high compared to the low statue in a defined period of time. The Arduino function analogwrite() is used to generate PWM signal of values between 0 and 255 which represents the 0 and the 100 duty cycle.
analogWrite(PIN, 64); 25 Duty Cycle or 25 of max speed
analogWrite(PIN, 127); 50 Duty Cycle or 50 of max speed
analogWrite(PIN, 255); 100 Duty Cycle or full speed

5.6.2 Control System Overview

The DC motor control system is to regulate the motors speed, ensuring it maintains a specific RPM (Revolutions Per Minute) through a closed-loop control approach. This system aims to achieve precise speed control by continuously adjusting the motor's operation based on feedback.

Key Components

The system comprises several key components:

- **Encoder sensor**: Measures motor speed by counting pulses generated by the rotating motor shaft.
- **Arduino board**: Processes this data to compute RPM and runs a PID controller to adjust motor speed.
- **DRV8833 motor driver**: Converts the Arduino's PWM signals into a higher voltage suitable for driving the motor and controls its direction.
- **JGA25-370 DC motor**: Driven by the DRV8833 based on Arduino signals.
- **Power Supply**: Provides the necessary voltage for the motor, encoder, and driver circuitry.

Operation

The inputs to the system include the Encoder Sensor, which provides pulse data for RPM measurement, and the Setpoint Block, which specifies the target speed. Outputs include the PWM Signal, which controls motor speed, and the adjusted Motor Speed and Direction, managed by the DRV8833 driver to drive the JGA25-370 DC Motor.
The control mechanism consists of a PID Controller that adjusts the PWM signal based on the difference between the actual speed and target RPM. Then using the feedback loop we can track the error variation for real time adjustment then the Arduino processes this data to modify the PWM output, which is then sent to the DRV8833 motor driver. Ensuring alignment with the target RPM and promptly correcting any deviations. The Arduino board PWM signal is limited to 5V, meaning it can't directly generate the higher voltage needed to fully power the motor. Even with a full-duty cycle , the motor driver can only apply a fraction of the total available voltage. This creates a saturation, where the system can't increase the motor speed beyond a certain point, even if the PID controller demands it. As a result, the motor may not reach the desired speed, especially at higher setpoints, leading to steady-state errors and performance limitations.

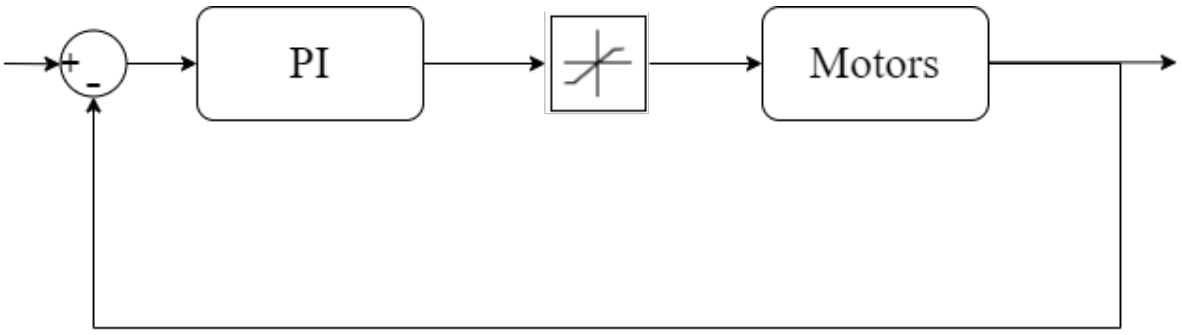


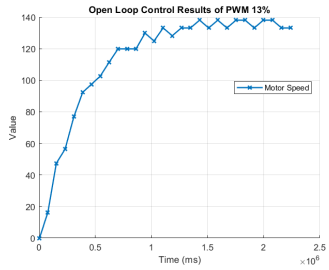
Figure 5.2: Diagram of the Motor Speed Control System

5.6.3 Tuning and Adjusting PID Controllers

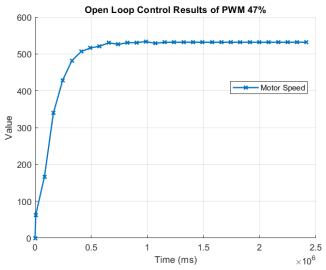
Manual iterative PID tuning of robot DC motors entails adjusting the proportional, integral, and derivative gains through empirical methods rather than relying on a mathematical model of the system. This process involves systematically varying PID parameters and evaluating the motors response to optimize performance metrics such as stability and responsiveness. In the absence of a precise system model, the iterative approach allows for empirical refinement of control parameters. Previous research [20], [21] on system identification and PID tuning provides valuable theoretical and practical insights, which can inform and streamline the manual tuning process, facilitating more efficient achievement of desired motor performance outcomes.
In some cases, a PID controller can be simplified to a PI, PD, P, or I controller by excluding certain actions. The PI controller omits the derivative action, which can be sensitive to measurement noise. With just Proportional and Integral gains, the focus is on balancing immediate response and long-term accuracy. To determine the optimal settings, it is necessary to first evaluate the system's response in an open-loop configuration, tuning the Proportional gain (Kp) initially, and then adjusting the Integral gain (Ki). Finding the right balance between these gains involves iterative testing to ensure the system achieves a quick, stable and accurate performance.
The PI controller tuning is explained through the next results and graphs.

Results of Open Loop Control

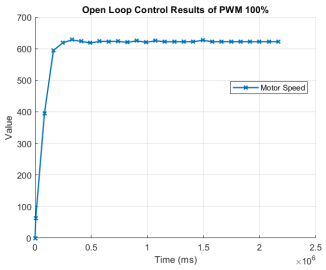
In an open-loop DC motor control system using an Arduino, the inputs are the desired speed settings and the PWM signal generated by the Arduino. The output is the motor's speed, which is adjusted based on the PWM signal. Since there is no feedback on the actual motor speed, the system assumes the output will match the input settings, but it cannot compensate for variations in load or performance [21].



(a) Duty cycle of 13%



(b) Duty cycle of 47%



(c) Duty cycle of 100%

Figure 5.3: Results of the Open Loop Control System

The graphs in the Figure (5.3) represent the motor speed responses under open-loop control for different Pulse Width Modulation (PWM) duty cycles, specifically at 13%, 47%, and 100%.

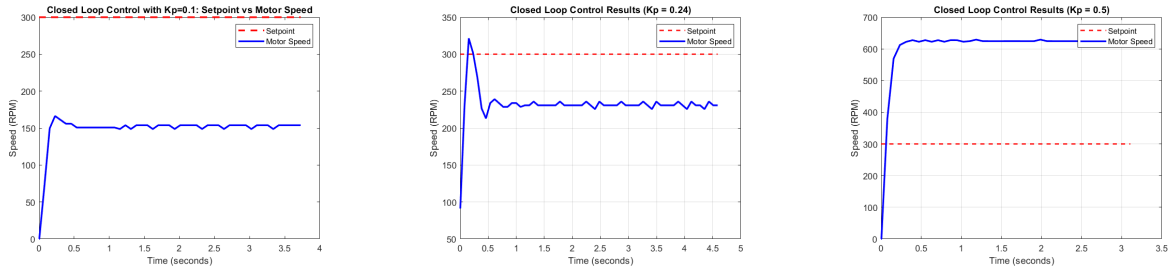
- Graph 1: PWM = 13%
  The motor speed gradually increases and stabilizes around 130 RPM. As the motor accelerates, there is a slow rise in speed, with a small irregular changes after reaching its peak before it stabilizes. This lower final speed is attributed to the small PWM duty cycle, where only 13% of the total possible power is applied to the motor. Consequently, the reduced power supplied to the motor results in a lower stabilized speed.

- Graph 2: PWM = 47%
  The motor speed increases quickly and stabilizes around 500 RPM, significantly higher than the previous case. It accelerates much faster and reaches a stable speed with minimal fluctuations. At a moderate PWM value, the motor achieves higher speed, and the control signal is strong enough to maintain stability.

- Graph 3: PWM = 100%
  The motor speed rises rapidly and stabilizes at its maximum value of around 600 RPM. It demonstrates the fastest acceleration and highest final speed, reaching its peak quickly and remaining stable without noticeable fluctuations. With 100% PWM, the motor operates at full power, providing a near-instantaneous response and high stability.

- Observations and Conclusion :
  The motor speed increases in relation to the PWM percentage, with higher duty cycles resulting in higher speeds, generally in a linear fashion in open-loop control. After a brief period of acceleration, the motor stabilizes with minimal fluctuations. However, open-loop control lacks adaptability to external factors like load or resistance, making it less flexible than closed-loop control. Overall, while the PWM duty cycle effectively controls motor speed, the absence of feedback means the system cannot adjust for variations in external conditions.

Results of Proportional Gain Tuning



(a) Kp = 0.1 (b) Kp = 0.24 (c) Kp = 0.5

Figure 5.4: Results of Kp Tuning

The provided plots in the Figure (5.4) show the motor speed response under closed-loop control for different proportional gain values (K_p). The setpoint is 300 RPM, and we observe how the motor reacts for different values of K_p.

- Graph 1: $K_p = 0.1$
  The motor speed is low and stabilizes around 150 RPM, well below the 300 RPM target. The low K_p value results in weak corrective action, causing a slow response and a high steady-state error. The system fails to meet performance requirements. In conclusion, $K_p = 0.1$ is too low to achieve the setpoint, leading to insufficient control action and a large steady-state error.

- Graph 2: $K_p = 0.24$
  The motor speed shows some initial overshoot but stabilizes closer to the 300 RPM setpoint, leveling out around 225-250 RPM, which is below the target. Despite this improvement, there is still a noticeable steady-state error, meaning the motor does not fully reach the setpoint. This indicates that proportional control alone is not enough to fully eliminate the error, suggesting the need for an integral term for complete correction over time.

- Graph 3: $K_p = 0.5$
  The motor speed accelerates quickly and overshoots the 300 RPM setpoint, reaching around 600 RPM and remaining there, unable to return to the target. Where the proportional control signal is too large, preventing the system from stabilizing at the desired speed. The motor ultimately stabilizes at a much higher speed than intended, indicating instability caused by excessive proportional gain. In conclusion, a $K_p = 0.5$ is too large for this system, leading to overshoot and saturation of the control signal.

- Observations and Conclusion :
  At $K_p = 0.1$, the system is stable but too slow and unable to reach the setpoint, resulting in a large steady-state error. With $K_p = 0.24$, the motor speed approaches the setpoint but still shows a steady-state error, indicating the need for an integral

term to fully correct it. At $K_p = 0.5$, the system overshoots significantly and becomes unstable. As a conclusion The low K_p results in slow performance, medium K_p improves response but requires further correction, and high K_p causes instability due to excessive control effort.

Results of Integral Gain Tuning

(a) Ki = 0.01 (b) Ki = 1.2 (c) Ki = 1.3

Figure 5.5: Results of Ki Tuning

The results of the Figure (5.5) are from the three motor speed responses with varying integral gain (K_i) show how changes in K_i affect the system's ability to maintain the motor's speed close to the 300 RPM setpoint and enhances accuracy but it must be balanced to avoid instability.

- Graph 1: $K_p = 0.24, K_i = 0.01$
  After the initial spike, the motor speed stabilizes around 225-250 RPM, below the desired 300 RPM, indicating a steady-state error. The small integral gain ($K_i = 0.01$) is insufficient to correct this error over time. Although the system is relatively stable with low oscillation amplitude, it remains inaccurate and unable to reach the setpoint, showing that the low K_i fails to eliminate the steady-state error.

- Graph 2: $K_p = 0.24, K_i = 1.2$
  With $K_i = 1.2$, the steady-state error is significantly reduced, and the motor speed oscillates closely around the 300 RPM setpoint. There is a slight initial overshoot to around 330 RPM, but the system quickly compensates and stabilizes near the target. While small oscillations indicate minor instability, the system performs well overall, reaching the setpoint with minimal error.

- Graph 3: $K_p = 0.24, K_i = 1.3$
  With $K_i = 1.3$, the motor speed far exceeds the 300 RPM setpoint, stabilizing around 600 RPM, indicating saturation. The controller overcompensates, leading to excessive motor speeds and an inability to bring the speed back to the setpoint. This demonstrates that $K_i = 1.3$ is too high, causing significant overshoot and saturation.

- Observations and Conclusion :
  The key observations show that with low $K_i = 0.01$, the system is stable but has a large steady-state error, keeping the motor speed below the setpoint. At moderate $K_i = 1.2$, the system performs best, minimizing the steady-state error with only slight oscillations near the setpoint. However, with high $K_i = 1.3$, the system saturates, causing the motor speed to overshoot and stabilize far above the target. In conclusion, tuning K_i is crucial for balancing steady-state error and stability, with $K_i = 1.2$ offering optimal performance by minimizing error without causing instability or overshoot.

5.6.4 PID Control Challenges and Solutions

During our robotics project, we faced challenges with PID control, including issues like windup saturation, varying motor gains, and sliding on different surfaces. We implemented several solutions to address these problems and improve the robots performance.

Windup Saturation

Problem: Windup saturation occurs when the integral term of the PID controller accumulates excessive error during periods when the motor is overloaded or stuck, leading to a situation where the controllers output becomes saturated. This often happens when the motor experiences uneven loading, with one side bearing more load than the other, causing the robot to be stuck or behave erratically. Additionally, startup errors can compound this issue by introducing additional errors into the system right from the beginning.
Solution: To mitigate windup saturation, the integral term of the PID controller is conditioned. This involves:

- **Conditioning the Integral Term:** Initiate the integral term only after a certain period or threshold, preventing it from building up excessively at startup.
- **Limiting the Integral Term:** Cap the integral term to a maximum value to avoid saturation.
- **Error Reset:** Reset the error term when transitioning between different control loops (e.g., when shifting from regulating the position to the yaw angle). This prevents the accumulation of errors that are irrelevant to the current control loop.

Variation in Motor Gains

Problem: Variations in motor characteristics and weight distribution can cause discrepancies in the effective gains of the PID controller across different motors. This challenge is exacerbated by differences in motor design and the distribution of the robots weight among the four motors.
Solution: To address this issue:

- **Unified PID Settings:** Find a set of PID gains that work reasonably well across all motors, even if they are not the optimal configuration for each individual motor. This ensures that the robot maintains consistent performance despite the variations in motor characteristics.

Surface Changes and Sliding

Problem: Different surfaces can cause the robot to slide, impacting its stability and control. This variation in traction necessitates testing on a variety of grounds to understand how surface changes affect the robot's performance.
Solution: To improve performance on varying surfaces:

- **Anti-Slide Mecanum Wheels:** Replace the 3D printed mecanum wheels with anti-slide versions. These wheels are designed to provide better traction and reduce the impact of surface variations on the robots movement.
- **Gain Optimization:** Recalibrate PID gains specifically for the new anti-slide wheels to ensure optimal performance on different surfaces.

5.6.5 Conclusion

In conclusion, the motor control system successfully regulates speed with the tuned PID parameters of $K_p = 0.24$ and $K_i = 1.2$. These settings provide balanced responsiveness and error correction, ensuring the motor maintains the desired speed effectively despite the Arduino's PWM limitations and potential saturation at higher setpoints.

5.7 Yaw Angle Control

5.7.1 Yaw Angle Control in Mecanum-Wheeled Robots

Mecanum wheel robots are versatile machines that can move in any directionforward, backward, sideways, and diagonally. To achieve precise movement and stability, it's crucial to control the robot's yaw angle. The yaw angle measures how much the robot is rotating around its vertical axis. Properly managing the yaw angle ensures the robot can turn smoothly and stay on course. This chapter explains the importance of yaw angle control, how it works, and how it enhances the robots performance in various situations.

What is Yaw Angle and Why Is It Important?

The yaw angle represents the robot's orientation around its vertical axis. It is measured in degrees and indicates how much the robot is turning. Controlling the yaw angle is essential for making accurate turns, aligning with targets, and preventing unwanted

drifting. When yaw angle is well-controlled, the robot can execute precise movements, such as turning in place or navigating tight spaces, without losing balance or direction. Without proper control of the yaw angle, the robot may experience wobbling or drifting, making it difficult to follow a precise path. This control is particularly important in applications like warehouses or hospitals, where the robot must move accurately in confined or busy areas.

How Yaw Angle Control Works

To control the yaw angle, the robot uses feedback from sensors such as gyroscopes or IMUs (Inertial Measurement Units). These sensors measure the current yaw angle and compare it to the desired yaw angle [22]. If there is a discrepancy, the robots PID controller adjusts the speeds of the mecanum wheels to correct the difference. The wheels are adjusted in coordination to create the right rotational force.

Benefits of Yaw Angle Control

Controlling the yaw angle provides several advantages that enhance the robot's overall performance:

- **Precise Turns**: The robot can make exact turns or spin in place without losing accuracy.
- **Stable Navigation**: It helps prevent wobbling or drifting, especially during complex movements.
- **Handling Obstacles**: When encountering uneven surfaces or obstacles, yaw angle control helps the robot stay on course and adjust its orientation.
- **Quick and Smooth Movements**: The robot can change directions quickly and smoothly, which is crucial in fast-paced environments like automated warehouses.

Resistance to Perturbations

Robots often encounter external forces, such as bumps or obstacles, that can affect their movement. Yaw angle control helps the robot detect these changes and adjust its wheel speeds to correct its orientation. This self-correction makes the robot more reliable in dynamic environments, allowing it to maintain stability even when facing unexpected disturbances.

5.7.2 Implementation

The asservissement system features two regulation loops designed for precise control of the robot's movement and orientation. The first loop, involving an Arduino and MPU6050

IMU, regulates the robot's angular position. The input for this loop is the desired angular position, and the output is the angular velocity needed to achieve that position. The second loop uses motor speed encoders to monitor and adjust the actual speed of the motors [23]. Here, the desired motor speed is the input, and the output is the measured motor speed. Together, these loops ensure the robot accurately follows its intended path and maintains stable movement by adjusting motor speeds based on both position and speed feedback.

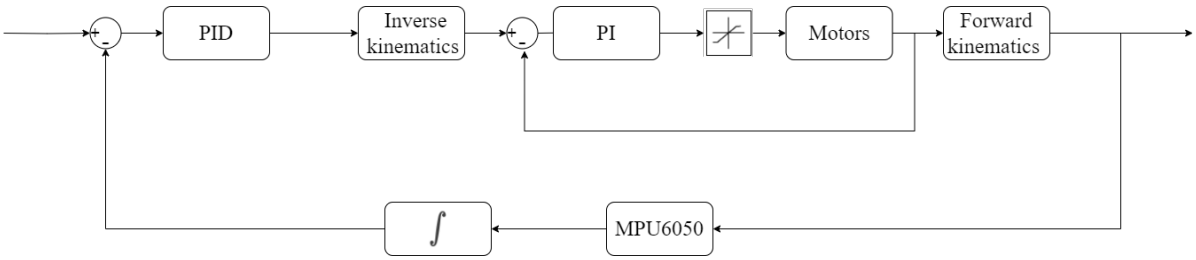


Figure 5.6: Yaw Angle Regulation System

5.7.3 MPU6050 Setup for Yaw Rate Measurement

Initialization: Configure the MPU6050 to measure yaw rate. The gyroscope in the MPU6050 outputs raw data representing rotational rates around the X, Y, and Z axes. For yaw rate control, focus on the Z-axis data.
Data Conversion: Convert the raw gyroscope data to angular velocity. The MPU6050s sensitivity setting is configured via the GYRO_CONFIG register, with values of ś250, ś500, ś1000, or ś2000 degrees per second (dps). The conversion formula is:

$$\text{Angular Velocity (dps)} = \frac{\text{Raw Gyro Value}}{\text{Sensitivity Scale Factor}}$$

For example, with a sensitivity scale factor of 131 for ś250 dps and a raw Z-axis value of 2048:

$$\text{Yaw Rate (dps)} = \frac{2048}{131} \approx 15.6 \text{ dps}$$

5.7.4 Yaw Rate Calculation and Integration

Yaw Rate Measurement: Extract yaw rate data from the MPU6050.
Yaw Angle Integration: To compute the yaw angle from the yaw rate, integrate the yaw rate over time. The formula for updating the yaw angle is:

$$\text{Yaw Angle}_{\text{new}} = \text{Yaw Angle}_{\text{old}} + \text{Yaw Rate} \times \Delta t$$

Where:

- Yaw $\text{Angle}_{\text{new}}$ is the updated yaw angle.
- Yaw $\text{Angle}_{\text{old}}$ is the previous yaw angle.
- Yaw Rate is the current measured yaw rate.
- Δt is the time interval between measurements.

5.7.5 Calibration and Filtering

Calibration: To correct for biases in the gyroscope readings, calibrate the sensor by averaging the raw data when the sensor is stationary and subtracting this average from future readings. This offsets any inherent biases in the gyroscope data.
Filtering: Apply filtering techniques to enhance the accuracy of the yaw angle calculation. Filters like the Kalman filter reducing noise and improving the robustness of yaw angle estimation.
In our study, we used two filters for processing MPU6050 data:
The Complementary Filter combines accelerometer and gyroscope data to provide a stable and accurate estimate of angular position, minimizing drift over time [24].
The Weighted Moving Average (WMA) calculates the average of past values by applying specific weights to each value. These weights, which are manually set, determine the relative importance of each value in the average calculation. Typically, the sum of the weights equals 1 to ensure a proper average. To compute the WMA for n periods, where x_t represents the value at time t and w_t is the weight assigned to that value, using the formula:

$$WMA = \frac{w_1 \cdot x_1 + w_2 \cdot x_2 + \cdots + w_n \cdot x_n}{w_1 + w_2 + \cdots + w_n}$$

The key characteristics of WMA include its customizability, as you can adjust the weights to prioritize certain values over others.

5.7.6 PI Controller

The PI controller adjusts motor speeds to maintain the desired yaw rate by using two components: the Proportional Term (P), which corrects the error based on the difference between desired and actual yaw rates, and the Integral Term (I), which addresses cumulative errors over time. By adding the derivative term the system had reacted aggressively to This involves calculating the yaw error, using it to compute the control output, and adjusting the PWM signals to the DRV8833 motor driver. Encoder feedback further refines control by measuring actual motor speeds and adjusting PWM signals accordingly to ensure precise movement and stability.

5.7.7 Results of Yaw Angle Control

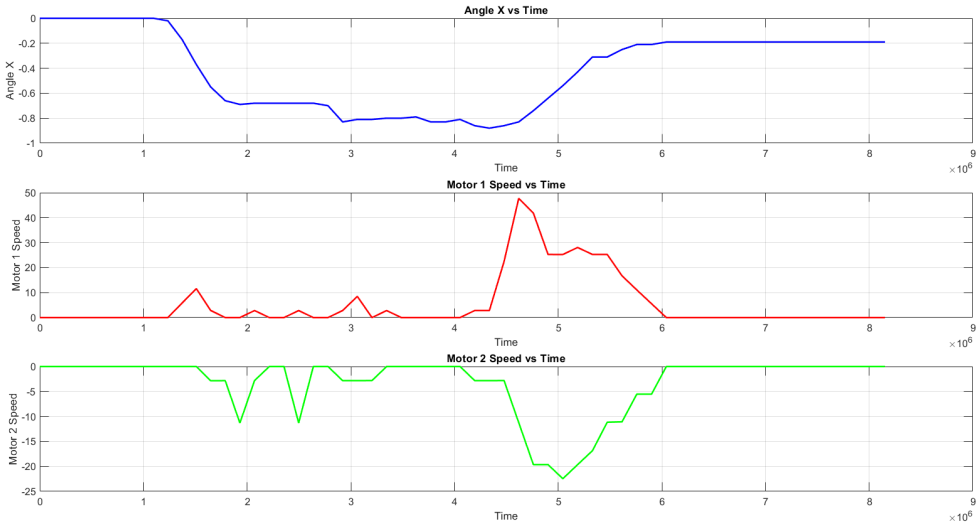


Figure 5.7: Evaluation of Yaw Angle Correction Results

In The set of graphs shown in the Figure (5.7), we observe the behavior of a closed-loop control system applied to a robot's yaw angle and the response of the motors to a perturbation. Here's an interpretation of the results, knowing that the goal is to correct the yaw angle and maintain stability:

- Top Graph: Angle X vs. Time
  The yaw angle starts at 0 radians, showing initial stability, but a perturbation at $t = 1 \times 10^6$ causes it to drop to nearly -1 radians, indicating a leftward rotation. The controller begins correcting at $t = 2 \times 10^6$, gradually bringing the angle back toward 0 radians, but it stabilizes just below 0 due to a threshold where the control regulation becomes inactive. In conclusion, the control system responds to the perturbation and successfully correct the yaw angle as we need. The angle does not go back to 0 because we pre-set an active interval for the control loop and a dead zone to not over correct.

- Middle Graph: Motor 1 Speed vs. Time
  Motor 1 remains inactive except some tries caused by the proportional term until $t = 2 \times 10^6$, when the integral term in the closed-loop controller activates, causing the motor speed to rise to around 40 RPM before decreasing back to 0. This increase in speed reflects the integral term's accumulation of error over time, driving the corrective effort to address the yaw angle error. As the error reduces, the motor speed decreases, demonstrating the integral term's effect in the control loop.

- Bottom Graph: Motor 2 Speed vs. Time
  Motor 2, like Motor 1, shows no initial activity but responds inversely due to the

differential drive system. At $t = 2 \times 10^6$, its speed becomes negative, reaching -20 RPM. Motor 2 slows down to help correct the yaw error, and after some oscillation, its speed returns to zero as the yaw angle stabilizes. In conclusion, Motor 2 acts in the opposite direction to Motor 1, adjusting its speed to assist in correcting the yaw angle and gradually returning to zero as the error is minimized.

- General Interpretation
  The closed-loop control system effectively detects and corrects the yaw angle perturbation, although it doesn't bring the angle completely back to 0 radians, due to the control dead zone. Motor 1 speeds up to correct the yaw, while Motor 2 do the same reaction in the backwards direction. The increase in motor speeds shows the activation of the integral term, which helps reduce steady-state error. After the correction, the system stabilizes the yaw angle within a small margin, and motor speeds return to zero, indicating the control effort has ceased.

5.7.8 Managing Interference Between Yaw Angle and Motor Speed Loops

In the current control system, both the yaw angle regulation loop and the motor speed loop can interfere with each other if they operate simultaneously. When the yaw angle loop adjusts the motor speeds to correct the robots orientation, it can unintentionally conflict with the motor speed control, which tries to compensate for these changes as a disturbance. This interaction can cause the robot to continue moving, prompting further reactions from the yaw angle loop and creating a feedback cycle that disrupts the robots trajectory.
To address this, separated regulation has been chosen for cases involving large angular adjustments, where the yaw angle correction can have a significant impact on the trajectory. For small angular deviations, combined regulation is used, where the yaw angle loop introduces new setpoints to the motor speed control. However, when this combined regulation is applied to high-angle corrections, it can sometimes lead to abrupt changes in direction, which may destabilize the robot and disturb its movement. Therefore, careful tuning is required to maintain stability, especially during high-angle regulation.

5.8 Robot Position Control

Robot position control is essential for ensuring precise and accurate movement in automated systems. It allows a robot to reach its desired location while compensating for errors caused by factors like external disturbances or system dynamics. The primary purpose of position control is to maintain stability, avoid overshoot, and accurately guide the robot along its intended path [25] [26]. This is especially important in tasks requiring high precision, such as automated assembly, navigation in crowded spaces, or handling delicate objects, where precise positioning directly affects performance and efficiency.

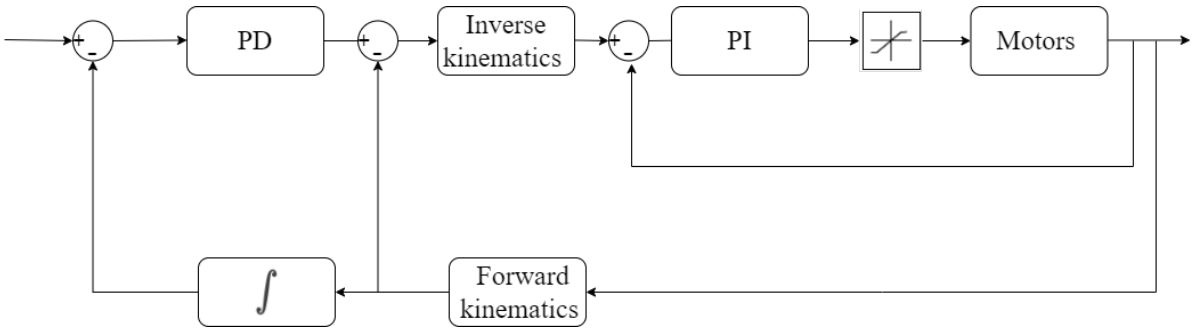


Figure 5.8: Structure of the Position Control System

5.8.1 Position Control Using a PD Controller

The PD controller controls the robot's position by calculating the velocity needed to reach the target, based on the position error (the difference between the current and desired positions). The derivative term stabilizes the robot's motion by reducing overshoot and preventing oscillations, reacting to the rate of change in the position error to ensure smooth movement. The output of the PD controller is the desired velocity, which serves as the input for the speed control loop.

5.8.2 Why PD Control is Used in Robotics

In robotics, **PD control** is commonly preferred over **PID control**, especially for position control, for several reasons:

Preventing Overshoot and Instability

The integral term accumulates past errors, potentially leading to aggressive corrections that cause overshoot and instability, which is undesirable for precision in robotic movements.

Handling Dynamic Set-points

Robots often have continuously changing target positions. PD control allows for quick reactions to these changes, while the integral term, which focuses on eliminating small steady-state errors, becomes less important when speed and responsiveness are prioritized.

Small Steady-State Error Without I Term

With highly accurate kinematic models, robots often exhibit minimal steady-state errors without the need for the integral term. The derivative term smooths the approach to the target, reducing error effectively.

Faster Response

The integral term can slow down the system by focusing on minor, long-term errors. In robotics, quick and accurate responses are more critical than eliminating small residual errors, making PD control more suitable.

Simpler Tuning

PD control is easier to tune than PID because the I term adds complexity and sensitivity, which can lead to instability. By omitting the integral term, tuning becomes simpler, and the system is more robust.

Natural Stabilization Due to Friction

In many robots, mechanical friction in joints or wheels naturally reduces small errors. This makes the I term unnecessary, as the system tends to stabilize on its own.

In conclusion, PD control offers a faster, simpler, and more stable solution for position control in robots, making it a better fit for most robotic applications where speed and precision are key.

5.8.3 Speed Control Using Velocity Feedback

Once the desired robot velocity is computed by the PD controller, it is sent to the speed control loop. Here, the desired robot speed is compared against the actual speed of the robot, which is measured using sensors or encoders on the wheels or motors. The difference between the desired speed (from the PD controller) and the actual speed (from encoders) is calculated, and this *speed error* is used to adjust the motor speeds to ensure the robot moves at the correct velocity.

5.8.4 Inverse Kinematics for Motor Speed Calculation

The robot have multiple wheels and actuators, it is necessary to convert the desired robot velocity into individual motor velocities. This is done through *inverse kinematics*, which takes into account the robot's configuration (e.g., differential drive, omnidirectional, etc.) and calculates the appropriate speeds for each motor.

5.8.5 Motor Speed Setpoints and Regulation Loop

The motor velocity commands calculated by the inverse kinematics are sent to the *motor control loop*, where each motor's speed is regulated. The motor controllers typically use a low-level *PID loop* to regulate motor speed, ensuring that each motor reaches and

maintains its desired setpoint velocity. This motor speed regulation loop adjusts the voltage or current supplied to the motors to precisely control their rotation.

5.8.6 Forward Kinematics for Robot Velocity Calculation

As the motors spin, their actual speeds are measured and converted back into the robot's overall velocity using *forward kinematics*. This process transform the individual motor speeds into the actual linear and angular velocities of the robot. The forward kinematics calculation considers the geometry of the robot (e.g., wheel radius, wheel separation, etc.) to compute the true velocity of the robot based on the motor speeds.
The actual robot velocity calculated from forward kinematics is then used in two critical ways:

- **Speed Feedback for Speed Control**: This velocity is compared to the desired velocity (from the PD controller) in the speed control loop to continuously adjust the motor speeds.
- **Position Feedback for Position Control**: At the same time, the robot's velocity is integrated over time to calculate the robot's position. By summing the velocity over small time intervals, the robot's position is continuously updated. This position feedback is then sent back to the PD controller, completing the control loop.

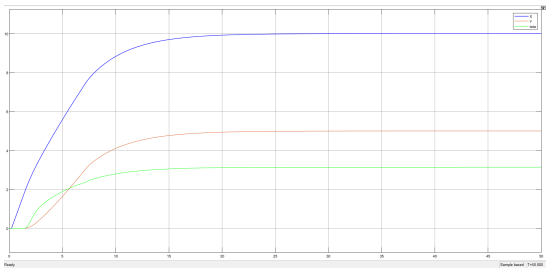
5.8.7 Position Integration and Closing the Loop

The robots actual position, obtained by integrating the velocity, is fed back to the *PD position controller*, where it is compared against the desired position. The difference between the actual position and the desired position generates a new position error, and the PD controller updates its output accordingly. This process is repeated continuously, forming a closed-loop system that ensures the robot maintains accurate control over its position.

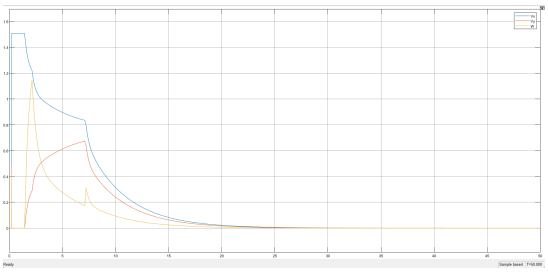
5.8.8 Results of the Controller

We conducted a simulation to explore and test all possible scenarios, ensuring a thorough evaluation of the controller's reliability. This approach allowed us to analyze the system's performance under various conditions. The details of these simulations and their outcomes are extensively explained in the masters thesis, After conducting several iterative tests, we identified that the suitable values for the PD controller gains are Kp = 1 and Kd = 2. These gains were found to produce the desired system behavior, allowing the controller to effectively compute the necessary velocity for reaching the target position while maintaining stable control over the robot's movement. and these are the results:

Position Control Without Perturbation



(a) Position Variation



(b) Speed Variation

Figure 5.9: Results of Position and Speed Variation Without Perturbation

These graphs represent in the Figure (5.9) are the simulation of a mobile robot equipped with mecanum wheels under position control, with the system being analyzed using a Proportional-Derivative (PD) controller. Below, we interpret each set of graphs and discuss how the PD controller influences system performance.

- First Graph: Position Control (X, Y, θ)
  - **X (Blue curve):** The robots movement in the X direction shows a smooth rise, stabilizing around 10m. The curve follows a classic first-order system behavior with no significant oscillations, indicating the robot smoothly approaches its target position in the X direction.
  - **Y (Orange curve):** The movement in the Y direction stabilizes around 5m, following a stable path but at a slower rate than the X direction.
  - θ **(Green curve):** The Z-axis (which might represent orientation or height) rises to about 3.14rad and stabilizes, indicating that the robot achieves the required Z-axis position or angle change.

  Interpretation: The overall system behavior shows a stable response in each axis. The lack of overshoot or oscillations in the X, Y, and θ directions indicates that the PD controller is well-tuned, avoiding excessive overshoot while ensuring the robot reaches its target. The rise times for X, Y, and θ differ, showing the robot takes varying amounts of time to stabilize in each direction.

- Second Graph: Speed Control (Vx, Vy, W)
  - **Vx (Blue curve):** The linear velocity in the X direction starts high, peaking around 1.4 units, and then decays exponentially as the robot approaches the target position. This indicates that the robot starts fast and gradually reduces its speed as it nears the target.
  - **Vy (Orange curve):** The Y velocity peaks later than the X velocity, indicating the robot takes more time to initiate significant movement in the Y direction. The velocity decreases similarly to Vx but with a delay and smaller

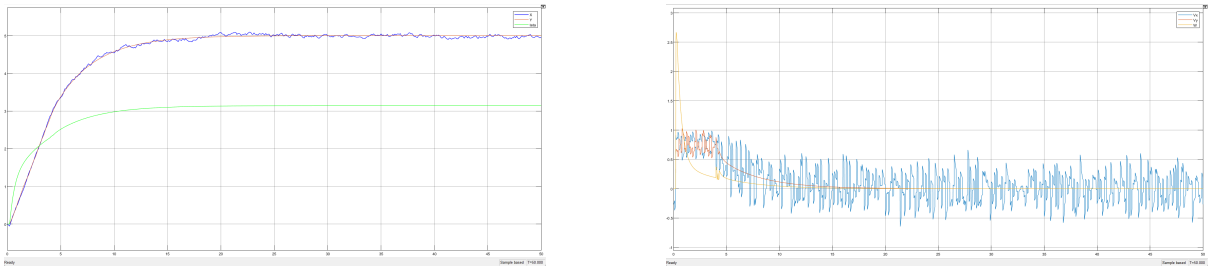
amplitude and this is because the high value of X setpoint compared to the Y.

- **W (Yellow curve):** The angular velocity peaks sharply before quickly settling to 0. This initial peak suggests the robot is correcting its orientation as it aligns to the desired direction during movement.

Interpretation: The speed control graph shows the PD controller adjusting velocity based on positional error. The robot starts with high speed to correct larger errors and gradually slows down as it nears the desired position.

- Conclusion
  The PD controller appears well-tuned for the system, achieving smooth stabilization without overshoot or oscillations. The rise times are appropriate, though slightly slower in the Y and θ directions, due the different set points. The position and velocity control graphs show that the PD controller provides stable and effective control over the robot's movement in all directions. The robot accelerates to correct the error and decelerates smoothly as it approaches the target. . Overall, the PD controller ensures the robot reaches its target position without overshoot, delivering smooth control.

Position Control with Speed Perturbations



(a) Position Variation (b) Speed Variation

Figure 5.10: Results of Position and Speed Variation With Perturbation

The provided graphs in the Figure (5.10) represent the position and speed control of a robot, with perturbations affecting the motor speed output during the control loop. Below is an interpretation of the graphs and key observations:

- Position Control (First Graph) **X, Y, and θ:** The first graph shows the robots position in the X and Y axes, along with its orientation θ.
  - The X and Y values demonstrate that the robot is successfully moving towards its target, with both position variables rising smoothly at first.
  - Theta increases at a slower rate, reflecting a slower angular change compared to the linear movements. This is consistent with how the robot moves, adjusting orientation as it approaches its target.

  - However, as the system faces a speed perturbation, the X values experience slight oscillations. These are caused by disturbances in the motor speed, showing how external factors affect position accuracy.

- Speed Control (Second Graph)
  The second graph plots the velocity of the robot in the X (Vx), Y (Vy), and angular velocity(W).

  - Initially, all velocities experience a peak, indicating the robot is accelerating towards the target. This is followed by a gradual deceleration as the robot approaches the setpoint.
  - Vx shows significant oscillations after around 5 seconds, indicating that motor speeds are being disrupted by perturbations in the control loop, due to external disturbances.
  - Vy and W also experience some small irregular changes, but these are less pronounced than the oscillations seen in Vx. The angular velocity (W) stabilizes quickly, indicating that the robots rotational speed is less impacted by the perturbation compared to its linear velocity in the X direction.

- Observations and Conclusion
  The control system initially performs well, with smooth position and speed behavior in the first 5 seconds, indicating proper functioning in the absence of disturbances. However, after this phase, the system experiences motor speed oscillations, particularly in the forward (Vx) direction, due to a perturbation and it is appeared only in the X axis because of the robot model where the perturbations get canceled in between because of the negative signs in the Vy and Wz equations as it shows in the following equations:

$$V_x = \frac{r}{4}(\dot{w}_{FL} + \dot{w}_{FR} + \dot{w}_{RL} + \dot{w}_{RR}) \tag{5.5}$$

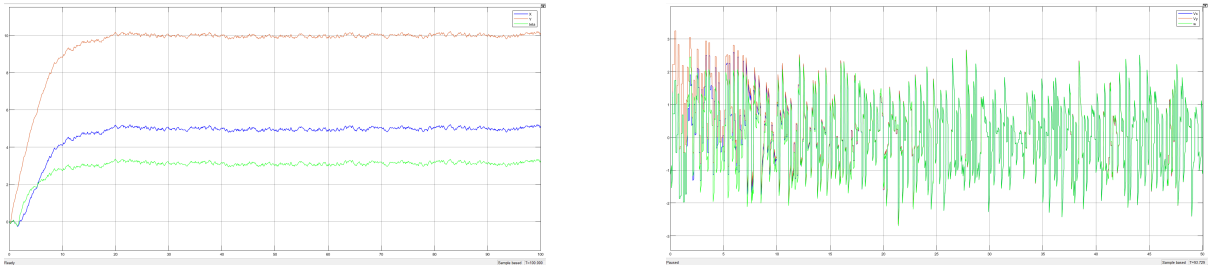
$$V_y = \frac{r}{4}(-\dot{w}_{FL} + \dot{w}_{FR} - \dot{w}_{RL} + \dot{w}_{RR}) \tag{5.6}$$

$$\omega_z = \frac{r}{4(L+W)}(-\dot{w}_{FL} + \dot{w}_{FR} + \dot{w}_{RL} - \dot{w}_{RR}) \tag{5.7}$$

  - V_x is the linear velocity in the x-direction.
  - V_y is the linear velocity in the y-direction.
  - ω_z is the angular velocity about the z-axis.
  - r is the radius of the wheels.
  - L and W are the length and width of the robot.
  - $\dot{w}_{FL}$, $\dot{w}_{FR}$, $\dot{w}_{RL}$, $\dot{w}_{RR}$ are the rotational velocities.

These oscillations cause minor fluctuations in the X, showing that the system's speed control struggles to compensate for the disturbance. The forward motion of the robot appears more sensitive to speed variations than its lateral or rotational movements.

Position Control with Speed Feedback Perturbations



(a) Position Variation (b) Speed Variation

Figure 5.11: Results of Position and Speed Variation with Feedback Perturbation

The graphs provided in the Figure (5.11) show the position and speed feedback during the robot's position control, with the control loop encountering a perturbation affecting the speed feedback. Below is an interpretation of the results:

- Position Control (First Graph)
  The position control graph displays the robot's X, Y, and θ as it follows the desired trajectory. The X, Y, and θ values rise steadily before stabilizing, but small oscillations appear after approximately 10 seconds. These oscillations indicate that, while the robot is reaching the target positions, the control system is experiencing minor disturbances.

- velocity Control (Second Graph)
  The speed feedback graph shows the robot's velocity in the X, Y, and angular directions (Vx, Vy, and W) as it responds to the control loop. They display a significant oscillations, particularly after the first few seconds, indicating that the feedback system is struggling to maintain stable velocities, due to perturbations in the control loop. Suggesting that the control system is less effective in managing noises or disturbances in the feedback loop.

- Observations and Conclusion
  The system is achieving the desired position and orientation, but it struggles with instability and oscillations due to feedback disturbances. These perturbations, particularly in the speed feedback (Vy and angular velocity W), affect the robot's overall control performance.
  While the robot reaches its target positions (X, Y, and *Theta*), the small oscillations indicate that the position control is not fully robust. Further tuning of the PD controller or speed control loop, as well as improvements in feedback filtering,

could help reduce these oscillations and improve both position and speed control stability.

5.8.9 Conclusion

As a result, the robot position control system effectively stabilizes the robot's motion and ensures precise movement toward the target. By continuously adjusting for the rate at which the robot approaches its destination, the system minimizes overshoot and prevents oscillations, leading to smoother deceleration near the target. This approach enhances both positioning accuracy and overall stability, making the system more efficient and reliable in dynamic environments.

5.9 Servo Motor Regulation in Robotic Manipulators

Servo motors play a crucial role in robotics, particularly for applications demanding precise control over position, speed, and torque. The MG996R servo motor, with its superior torque, speed, and precision compared to the MG90, is highly favored for tasks involving robotic arms. Its capability to handle heavier loads and deliver accurate control makes it ideal for high-performance tasks. In contrast, the MG90, while smaller and lighter, lacks the torque and precision required for advanced projects, often making it less suitable for demanding applications. Achieving smooth and precise movements is vital for enhancing system efficiency and accuracy in robotic arms.

5.9.1 MG996R Servo Motor Identification

Modeling the MG996R Servo Motor Mathematically

Modeling the MG996R servo motor involves understanding its dynamics and response to control inputs. And these are the steps we followed to identify the MG996R servo motor.

1. **Define the System**
    - **Inputs**: PWM signal (control input), typically ranging from 500ţs to 2500ţs.
    - **Outputs**: Angular position of the servo motor.
2. **Collect Data**
    - **Input Data**: Record the PWM pulse widths sent to the servo.
    - **Output Data**: Measure the resulting angular positions of the servo at various time intervals.

3. **Choose a Model Structure**

   - **Linear Time-Invariant (LTI) Models**: Suitable for simplicity, especially if the servo operates within its linear range.
   - **Nonlinear Models**: Consider if the servo exhibits nonlinear behavior at certain control inputs or conditions.

4. **Linear Model Identification**

   - **Transfer Function Model**
     1. **Determine Transfer Function Form**:

$$G(s) = \frac{K}{Ts + 1} \tag{5.8}$$

        where K is the steady-state gain and T is the time constant.
     2. **Estimate Parameters**:
        - **Steady-State Gain (K)**: Measure the angular position for a constant PWM input. K is the ratio of the change in angle to the change in PWM input.
        - **Time Constant (T)**: Analyze the response time of the servo to a step input to estimate how quickly the servo reaches its new position.
     3. **Fit the Model**:
        - Use methods such as least squares to fit the estimated transfer function parameters to the experimental data.

Read, Write, and Analyze Data from Arduino

MATLAB support package for Arduino lets you write MATLAB programs that read and write data to your Arduino and access connected devices such as motors, LEDs, and I2C devices. Because MATLAB is a high-level interpreted language, prototyping and refining algorithms, and you can see results from I/O instructions immediately, without recompiling. MATLAB includes thousands of built-in math, engineering, and plotting functions that you can use for your Arduino programming [27].

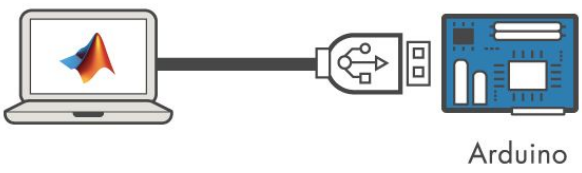


Figure 5.12: Connecting MATLAB to Arduino

- Interactively read and write sensor data without having to wait for code compilation.

- Develop and analyze algorithms using thousands of pre-built functions for signal processing, machine learning, mathematical modeling, and more.
- Rapidly visualize your data with MATLAB's extensive range of plotting options.

Results of Identification

We have used the MATLAB support package for Arduino to read the inputs signals and the angular position of the servo motor provided by the modified servo motor and in the following Figure (5.13) we can see Simulink blocks used in this process then in the Figure (5.14) we can see the resulting signal.

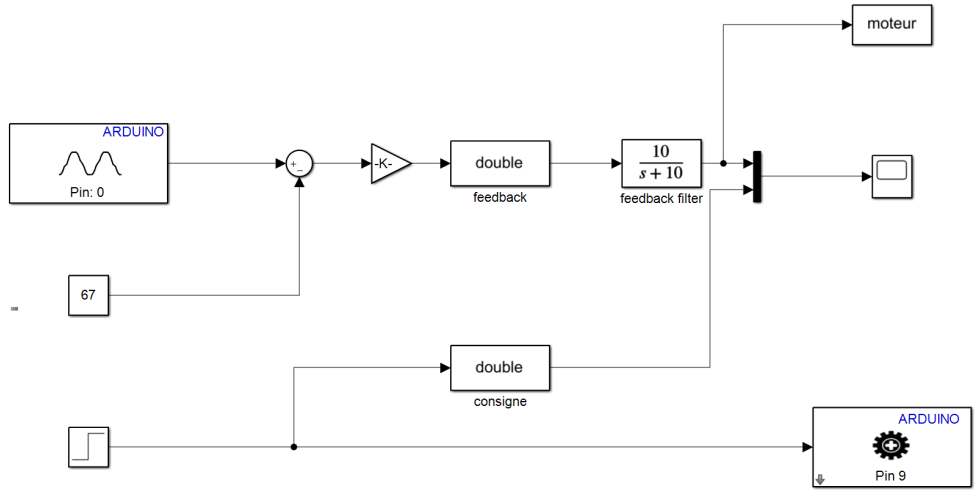
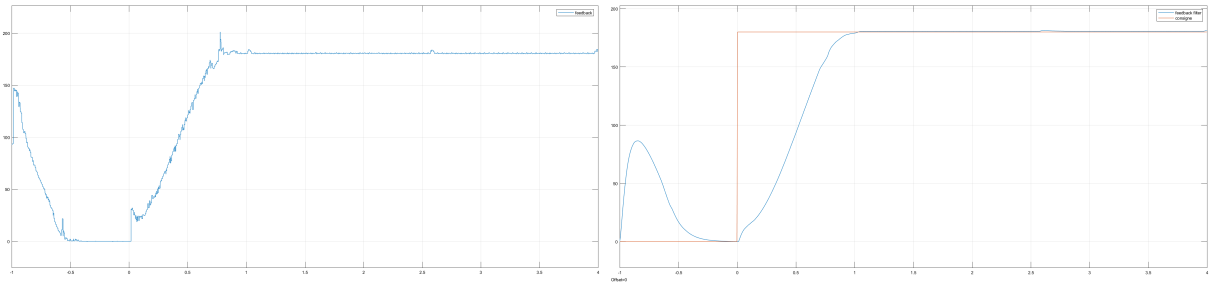


Figure 5.13: MATLAB Blocks for Reading and Filtering Servo Position



(a) Before Filtering (b) After Filtering

Figure 5.14: Evaluation of Servo Motor Step Response

- Filtration
  The provided graphs in the Figure (5.14) show the behavior of a servo motor subject to noise and then the filtered result after applying a filter. Lets analyze the details of each scenario and draw conclusions.
  - The First Graph (Servo Motor with Noise)
    The servo motor starts at 90 degrees, with the feedback signal initially stable.

It is then commanded to move to 0 degrees, causing a sharp drop in the command signal, during which the feedback shows some oscillations and noise before eventually reaching the 0-degree position. At time $t = 0$, the servo is commanded to move to 180 degrees, noise causes fluctuations during the movement. The servo ultimately reaches close to 180 degrees but with a small steady-state error of 1-2 degrees. Noise is evident throughout the movement, particularly during transitions and even after the system stabilizes.

- After Applying a Filter (Second Graph)
  The filtered feedback signal greatly reduces noise, making the servo's movement smoother and more stable. Transitions between positions, like from 90 to 0 degrees and 0 to 180 degrees, are much cleaner, with fewer disturbances. While there's still a small steady-state error of 1-2 degrees at 180 degrees, it's more noticeable now that the noise is reduced. The filter may have slightly slowed the feedback, but the servo still responds well to commands and reaches the target positions with minimal issues.
- Observations and Conclusion
  Before filtering, the servo motor experienced significant noise, especially during position transitions, leading to fluctuations in the feedback signal that impacted precision. After applying the filter, the noise was greatly reduced, resulting in smoother motion and more accurate feedback. However, a small steady-state error of 1-2 degrees remained, likely due to servo controller limitations, the remaining error suggests that further control changements may be needed to fully address this issue.

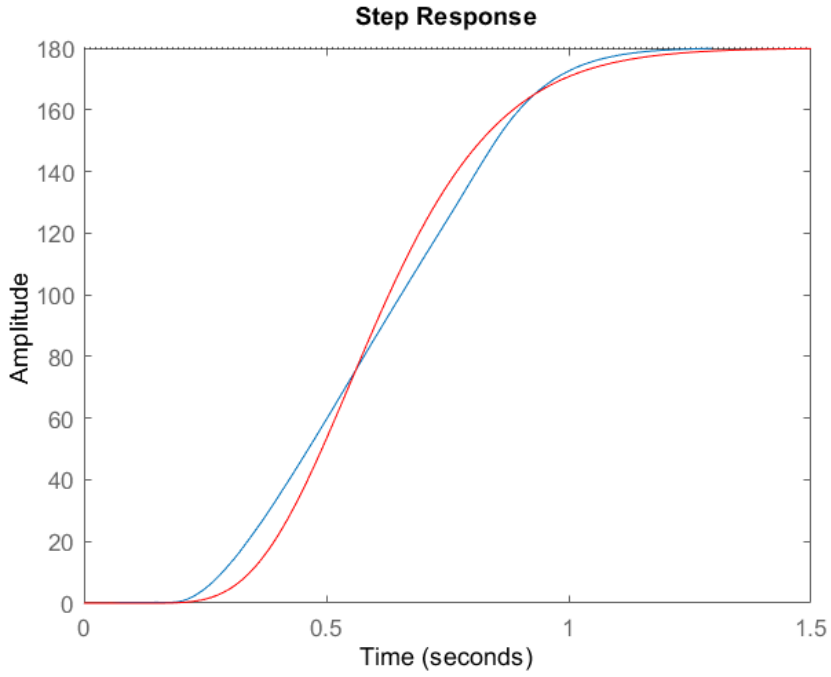


Figure 5.15: Results of the Strecj Identification Method

- The Strecj identification method results
  The graph in the Figure (5.15) shows the step response of a system (blue line) and its corresponding identified model (red line). The purpose of system identification is to create a mathematical model (the red curve) that closely mimics the behavior of the real system (the blue curve) in response to an input, in this case, a step input. In the case of our study the resulting system transfer function from the identification is :

$$g_2(s) = \frac{180 \cdot e^{-0.15s}}{(1 + 0.167s)^3}$$

  The blue curve rise smoothly and stabilize at 180 degrees in about 1.3 seconds without overshoot or oscillations, indicating stability. The red curve, showing the identified model's response, closely follows the system's behavior but with a slightly faster rise time at the start. This initial discrepancy suggests the model overestimates the system's speed early on, though both curves eventually converge at the same target value, showing that the model accurately captures the steady-state behavior. Overall, the model provides a good fit, though it may need refinement to better match the early dynamics.

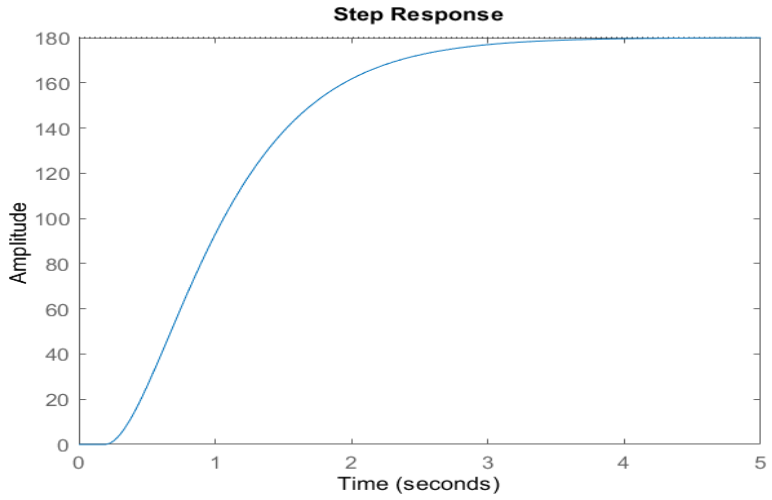


Figure 5.16: Results of the Broida Identification Method

- The Broida identification method results
  This graph shown in the Figure (5.16) represents the step response of a system identified using the Broida method, which is a system identification technique used to approximate the transfer function of a system based on its step response. In this case, the system is likely approximated using a first-order model with a time delay. In the case of our study the resulting system transfer function from the identification is :

$$H(s) = \frac{180 \cdot e^{-0.15s}}{0.5s + 1}$$

  The system's step response rises smoothly from 0 to the final value of 180 units

without overshoot or oscillations, indicating a stable first-order system. The minimal time delay suggests the system responds immediately, with a time constant of around 1.2 seconds, reaching 63% of the final value by that point. The system converges to the expected steady-state value of 180 units, confirming that the identified model accurately reflects the systems behavior. The absence of oscillations reinforces that the Broida method provides a good approximation of a first-order system, with minimal dead time and predictable, smooth response characteristics.

5.9.2 Challenges in Speed Control

The MG996R is primarily designed for position control, which can limit its effectiveness in precise speed regulation. Without modifications or external control, it may struggle to maintain consistent speeds, particularly in applications that demand slow, steady movements.
For these reasons we chose to implement a PID controller [28] and it's gain are tuned according to the identified model based on the strecj method.

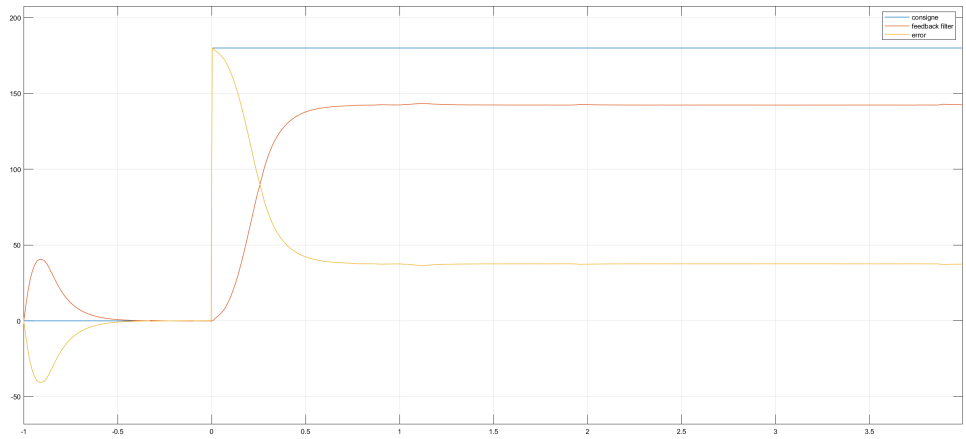


Figure 5.17: Implementation of PID Control on the Servo Motor

The provided graph in the Figure (5.17) depicts the performance of a servo motor controlled by an external PID controller, featuring three distinct lines.

- The consigne (blue line) represents the target or desired position for the servo motor, set at approximately 180 degrees and held constant throughout the operation.
- The feedback filter (orange line) shows the motors actual response under PID control. Initially, the motor moves quickly toward the target, demonstrating improved speed and stability. However, after a period of oscillation, the feedback stabilizes below the setpoint, indicating the presence of a steady-state error.
- The error (yellow line) visualizes the difference between the target and actual position. The error initially spikes as the motor adjusts but fails to reach zero, showing that the steady-state error persists.

The system shows improved stability and speed with the PID controller, as the feedback signal settles without further oscillations and responds quickly to changes. However, a significant steady-state error persists, meaning the feedback does not fully reach the desired setpoint of 180 degrees. While the controller enhances speed and stability, this steady-state error is problematic for precision applications, indicating the need for further tuning of the PID gains, especially the integral component, to improve accuracy.

5.9.3 PID Control for Servo Motor Speed

The MG996R servo motor features an internal control system that uses a proportional (P) controller. This controller continuously compares the servo's current position, as measured by a feedback potentiometer, with the target position set by the input PWM signal. It adjusts the motor's power output to reduce the error, which is the difference between the actual and desired positions. This straightforward control approach is effective for most standard positional control applications.
If an external PID (Proportional-Integral-Derivative) controller is added to the system, it introduces additional layers of control, creating a dual control loop, where both systems may try to adjust the position simultaneously. For example:

- The external P controller might try to correct the position, while the internal P controller is already doing the same. This can lead to overcorrection or even oscillations if the gains of both controllers are not properly tuned.
- The external D term could conflict with the internal P controllers response, as the derivative component anticipates future changes in error, possibly destabilizing the servos already tuned internal loop.
- The external I term might accumulate error corrections, which the internal P controller cannot account for, potentially causing sluggish response or oscillations.

The interaction between an external PID controller and the internal P controller of the servo requires careful calibration. If both controllers are not properly tuned to work in harmony, it may result in slower response times, increased oscillations, or even instability in the servos performance. To avoid these issues, the external PID controller should complement the internal P controller by addressing its limitations, such as accommodating longer time constants or providing finer precision, rather than merely duplicating its efforts and potentially causing conflicting adjustments.

5.9.4 Discretizing Cubic Trajectories for Speed Control

Polynomial Trajectory Control

Since PID gains values didn't fit the system we have to find an optimal gains which work in complementary with the servo motor control, this process could be time consuming for

these reasons we chose another control logic named the Cubic Trajectory Discretization for Speed Control [29].
Using cubic polynomial trajectories allows for smooth and continuous motion by interpolating between positions over time. This method is particularly beneficial in robotic arms, where smooth acceleration and deceleration prevent abrupt movements and improve system efficiency.

Cubic Polynomial Equation

The cubic polynomial trajectory equation is as follows:

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

Where:

- $\theta(t)$ represents the motor position at time t,
- a_0, a_1, a_2, a_3 are coefficients determined by initial and final conditions (positions, velocities, and accelerations).

Trajectory Discretization

By breaking the trajectory into discrete intervals, the motor follows a smooth and consistent path. This ensures steady control of speed and position, avoiding sudden movements, and providing precise motion throughout the entire operation.

Implementing the Polynomial Trajectory for Robotic Arms

- **Choosing Final Time and Step Sizes**
  Final time and step sizes are essential for smooth motion. The final time affects movement speed, while step size determines trajectory discreteness and motion smoothness.
- **Ensuring Smooth Transitions**
  Cubic polynomial control ensures gradual acceleration and deceleration, preventing sudden speed changes that could damage the motor.
- **Simulation and Real-World Implementation**
  Simulate cubic trajectory control to validate its effectiveness before real-world application, ensuring smooth and precise movement for robotic arms using the MG996R.

Chapter 6

Camera Vision

Camera vision, also known as computer vision, is a field of artificial intelligence that enables machines to interpret and make decisions based on visual data from the real world. It encompasses various processes, including image acquisition, preprocessing, feature extraction, and interpretation. The ultimate goal is to replicate the capabilities of human vision, allowing machines to understand scenes, recognize objects, and make decisions based on visual input.

Applications

- **Autonomous Vehicles:** Computer vision is essential in self-driving cars for object detection, lane detection, and traffic sign recognition.
- **Robotics:** In robotics, vision systems enable robots to navigate, recognize objects, and interact with their environment.
- **Medical Imaging:** Vision techniques are used to analyze medical images for diagnostics, such as detecting tumors in MRI scans.
- **Surveillance:** Camera vision systems are widely used for monitoring and security purposes, such as face recognition in CCTV footage.

Challenges

- **Variability in Lighting:** Changes in lighting conditions can drastically affect image quality and, consequently, the performance of vision algorithms.
- **Object Occlusion:** Objects in a scene may be partially hidden, complicating recognition and analysis.
- **Real-Time Processing:** Many applications require real-time processing of visual data, demanding high computational power and efficient algorithms.

6.1 Pinhole Model

The pinhole model provides a simple but powerful framework for understanding how cameras project 3D scenes onto 2D images. The model assumes that all rays of light pass through a single point (the pinhole) before striking the image plane. This model forms the basis of most computer vision algorithms dealing with image formation [30].
The next Figure (6.1) shows an explanation of the pinhole model.

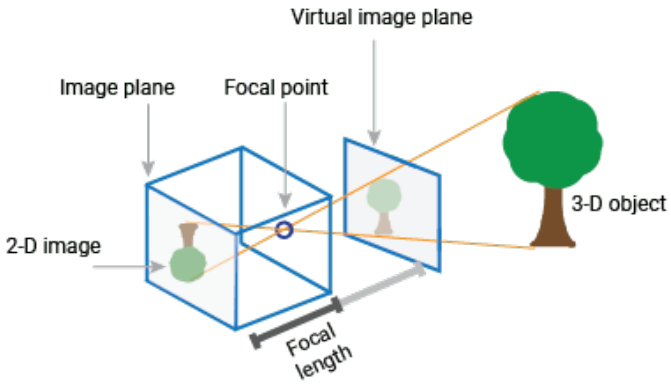


Figure 6.1: Overview of the Pinhole Model

6.1.1 Mathematical Model

The pinhole camera model is defined by the following projection equation:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

Where:

- $[X_w, Y_w, Z_w]$ are the world coordinates of a point.
- $[u, v]$ are the image coordinates on the 2D image plane.
- $\mathbf{K}$ is the intrinsic camera matrix.
- $[R \mid t]$ is the extrinsic matrix representing the camera's position and orientation.
- s is a scaling factor.

6.1.2 Practical Implications

The pinhole model provides a foundational understanding that helps in designing more complex models and algorithms. However, real cameras have lenses, leading to distor-

tions that the pinhole model doesn't account for, necessitating more advanced calibration techniques.

6.2 Camera Calibration

Camera calibration is a critical process in computer vision, where the goal is to determine the camera's intrinsic and extrinsic parameters. Calibration ensures that measurements and reconstructions from images are accurate [30], which is crucial for applications like 3D modeling, augmented reality, and robotics.

6.2.1 Intrinsic Parameters

The intrinsic parameters define the internal characteristics of the camera and include:

- **Focal Length** (f_x, f_y)**:** The distance between the camera lens and the image sensor, affecting the field of view.
- **Principal Point** (c_x, c_y)**:** The point where the optical axis intersects the image plane.
- **Skew Coefficient** α**:** Accounts for non-perpendicularity between the x and y pixel axes, which is often negligible in modern cameras.

6.2.2 Extrinsic Parameters

The extrinsic parameters describe the camera's position and orientation relative to the world coordinate system. They consist of:

- **Rotation Matrix** R**:** Describes the camera's orientation in space.
- **Translation Vector** t**:** Defines the camera's position in the world.

6.2.3 Calibration Process

- **Data Collection:** Images of a known calibration pattern (e.g., a checkerboard) are captured from different angles.
- **Feature Extraction:** Key points on the calibration pattern are detected in the images.
- **Optimization:** The intrinsic and extrinsic parameters are estimated by minimizing the difference between the observed image points and the projected points calculated using the pinhole model.

6.3 Extrinsic Parameters

Extrinsic parameters describe how the camera is positioned and oriented in the world. These parameters are essential for tasks like 3D reconstruction, where understanding the camera's viewpoint is necessary to accurately map the environment [31].

6.3.1 Rotation Matrix R

The rotation matrix R is a 3x3 matrix that describes how the camera's coordinate system is rotated relative to the world coordinate system. It can be decomposed into three rotation angles (roll, pitch, yaw) corresponding to rotations around the x, y, and z axes.

6.3.2 Translation Vector t

The translation vector t is a 3x1 vector that describes the camera's position relative to the world coordinate system. It indicates how far the camera is displaced along the x, y, and z axes from the origin of the world coordinate system.

6.3.3 Mathematical Representation

The transformation from world coordinates to camera coordinates is given by:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = R \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + t$$

Where $[X_c, Y_c, Z_c]$ are the coordinates in the camera frame, and $[X_w, Y_w, Z_w]$ are the world coordinates.

6.4 Intrinsic Parameters

The intrinsic parameters of a camera define the geometric properties of the camera's imaging process. They describe how the 3D world is mapped to the 2D image plane, independent of the camera's position and orientation [32].

6.4.1 Intrinsic Matrix K

The intrinsic matrix **K** is a 3x3 matrix that encapsulates the camera's internal characteristics:

$$\mathbf{K} = \begin{bmatrix} f_x & \alpha & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

- **Focal Length** (f_x, f_y)**:** Determines the camera's field of view and magnification.
- **Principal Point** (c_x, c_y)**:** The coordinates of the optical center on the image plane.
- **Skew Coefficient** α**:** Represents the skew between the image axes.

6.4.2 Calibration of Intrinsic Parameters

The intrinsic parameters are usually calibrated using images of known patterns (e.g., checkerboards), where the camera captures multiple views of the pattern, and the parameters are optimized to minimize the projection error.

6.4.3 Practical Considerations

- **Lens Distortion:** Real cameras often introduce distortions, especially near the edges of the image. These distortions are typically modeled using additional parameters, such as radial and tangential distortion coefficients.
- **Aspect Ratio:** In some cameras, the pixel aspect ratio (the ratio of the width to the height of a pixel) is not 1, which needs to be considered in the intrinsic matrix.

6.5 Transformation from World Coordinates (X, Y, Z) to Image Coordinates (U, V)

The transformation from 3D world coordinates to 2D image coordinates is a key operation in computer vision, underlying tasks such as rendering, scene understanding, and object detection. This transformation involves both the intrinsic and extrinsic parameters of the camera.

6.5.1 Mathematical Transformation

The transformation is expressed as:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} R & t \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

Where:

- $\mathbf{K}$ is the intrinsic matrix.
- $[R \mid t]$ represents the extrinsic parameters.
- $[X_w, Y_w, Z_w]$ are the world coordinates.
- $[u, v]$ are the image coordinates.
- s is a scale factor that depends on the depth of the point in the scene.

6.5.2 Steps in the Transformation

1. **World to Camera Coordinates:** The world coordinates $[X_w, Y_w, Z_w]$ are first transformed to camera coordinates using the extrinsic parameters:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = R \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + t$$

2. **Projection onto Image Plane:** The camera coordinates are then projected onto the image plane using the intrinsic parameters:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}$$

6.5.3 Challenges

- **Depth Information:** The transformation inherently loses depth information, making it difficult to recover 3D positions from 2D images without additional data.
- **Distortion Correction:** Real cameras often introduce distortions that must be corrected for accurate projection.

6.6 Transformation from Image Coordinates (U, V) to World Coordinates (X, Y, Z)

The reverse transformation, from image coordinates to world coordinates, is more complex and typically requires additional information. This process is fundamental in tasks like 3D reconstruction, where the goal is to recover the 3D structure of a scene from its 2D projections.

6.6.1 Mathematical Approach

To perform this transformation, the depth Z_c of the point must be known. Given the image coordinates $[u, v]$, the world coordinates $[X_w, Y_w, Z_w]$ can be recovered using:

$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} = R^{-1} \left(\mathbf{K}^{-1} \begin{bmatrix} su \\ sv \\ s \end{bmatrix} - t \right)$$

Where:

- R^{-1} is the inverse of the rotation matrix.
- $\mathbf{K}^{-1}$ is the inverse of the intrinsic matrix.
- t is the translation vector.

6.6.2 Techniques for Depth Estimation

- **Stereo Vision:** Uses two or more images taken from different viewpoints to triangulate the depth of points in the scene.
- **Structure from Motion (SfM):** Estimates depth by analyzing the motion of objects between consecutive frames in a video sequence.
- **LiDAR:** A sensor-based approach that directly measures the distance to objects, providing accurate depth information.

6.6.3 Challenges

- **Ambiguity:** Without depth information, multiple world points can correspond to the same image point, leading to ambiguity.
- **Accuracy:** Small errors in calibration or depth estimation can lead to significant inaccuracies in the recovered world coordinates.

6.7 Chessboard Camera Calibration with MATLAB Toolbox

Chessboard camera calibration is a crucial process in computer vision, used to estimate both the intrinsic and extrinsic parameters of a camera. These parameters are necessary for accurate 3D reconstruction, object tracking, or other vision-based applications. Below is a breakdown of the process and how it's done using the **MATLAB Camera Calibration Toolbox** [33].

6.8 Preparation: Capturing Chessboard Images

The calibration process starts by capturing several images of a flat chessboard pattern. The chessboard is typically a square grid with alternating black and white tiles. The corners of these tiles serve as feature points. To ensure accurate calibration, the chessboard is imaged from different orientations and distances, providing a variety of perspectives for the calibration process. More images generally lead to a better calibration, especially when taken from different angles and distances.
The following Figure (6.2) is an example of the chessboard images taken.

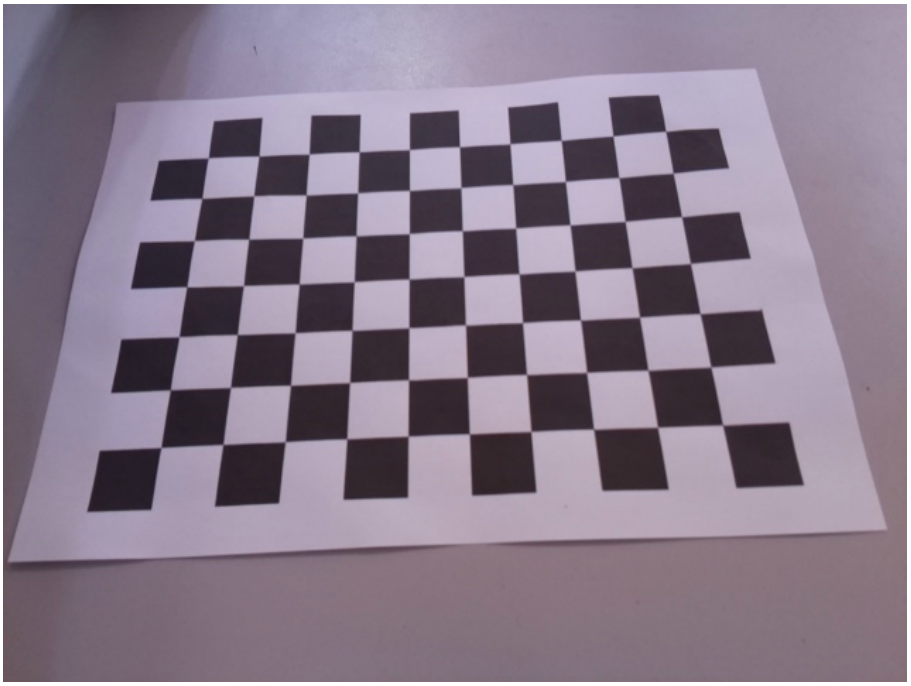


Figure 6.2: Chessboard Image

6.9 Loading Images into MATLAB

Once the images are captured, they are loaded into MATLAB through the **Camera Calibration Toolbox**. The toolbox can be accessed through MATLABs Apps tab or programmatically via scripts. The images should be in a common format like PNG or JPEG. The toolbox includes functionality to automatically detect chessboard corners in each image, which are the critical points for calibration.

6.10 Corner Detection and Point Extraction

The next step is automatic corner detection. MATLABs toolbox uses image processing algorithms to identify the intersection points of the chessboard tiles. These points are treated as known features because the distances between them are consistent and can be mapped in both the image and the real world. It is important to ensure good lighting and focus to maximize corner detection accuracy. If necessary, manual adjustments can be made to correct any misdetections.
The Figure (6.3) shown here is an example of corner detection and point extraction.

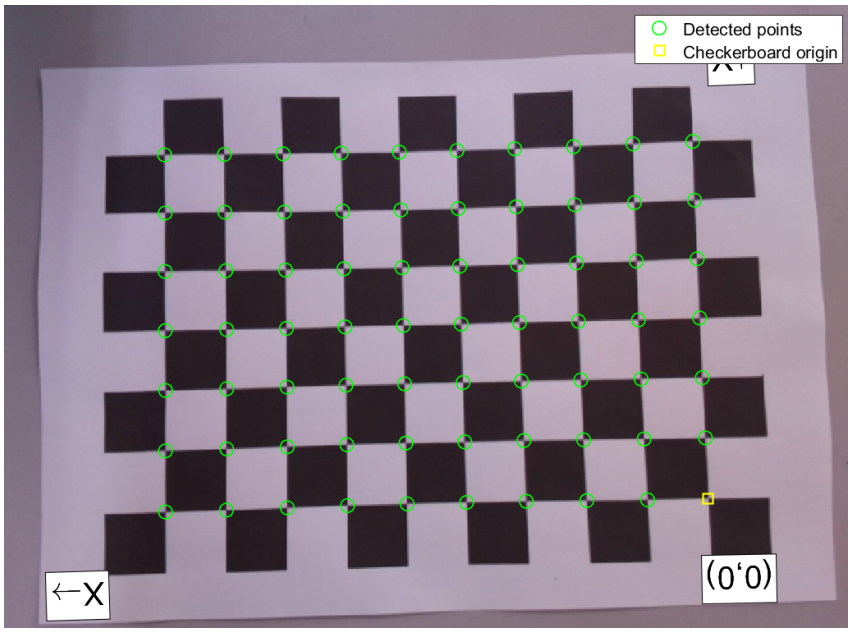


Figure 6.3: Corner Detection

6.11 Optimization and Reprojection Error

After estimating the intrinsic and extrinsic parameters, MATLAB performs an optimization to minimize the **reprojection error**, which is the difference between the observed corner positions and their projected locations based on the estimated camera model. The toolbox uses a nonlinear least-squares algorithm to refine the parameters, ensuring

accurate calibration. Low reprojection errors indicate a well-calibrated camera. The following Figure (6.4), (6.5) and (6.6) show the The mean error reprojection visualization, the camera position estimation and the chessboard position estimation.

Figure 6.4: Visualization of Mean Error Re-projection

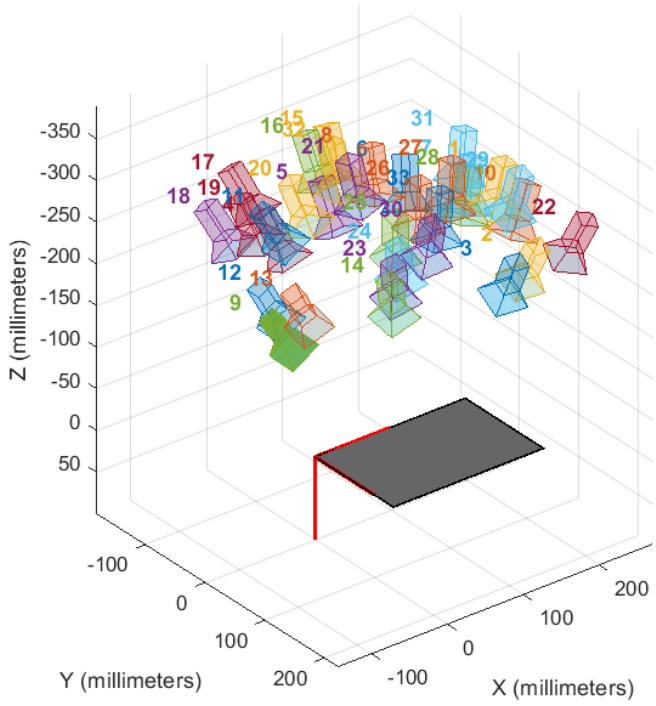


Figure 6.5: Camera Position Estimation

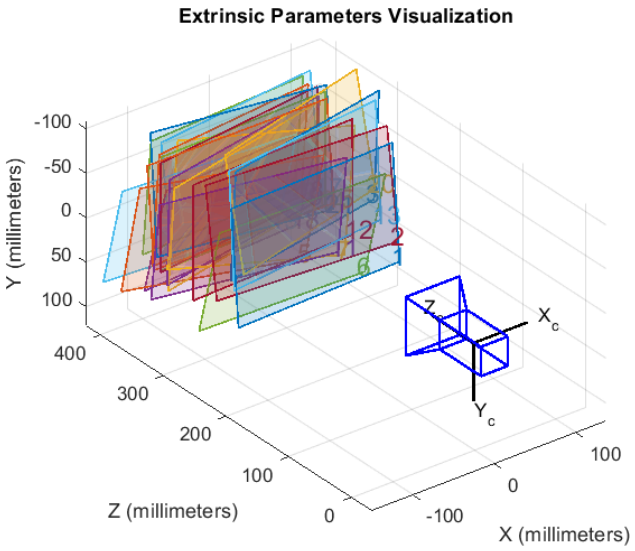


Figure 6.6: Extrinsic Parameter Estimation

6.12 Final Calibration Results and Usage

Once the calibration process is completed, MATLAB provides a camera model that includes the estimated intrinsic and extrinsic parameters, along with lens distortion coefficients. This calibrated model can be used to correct distorted images, project 3D points onto the 2D image plane, or map 2D image points back to 3D world coordinates.
For our case of study these are the calibration results:

Intrinsic Matrix K

- Focal lengths: $[f_x, f_y] = [504.3079, 505.2568]$
- Principal point: $[c_x, c_y] = [320.6196, 241.4465]$
- Skew: $s = 0$

So, the intrinsic matrix K is:

$$K = \begin{bmatrix} 504.3079 & 0 & 320.6196 \\ 0 & 505.2568 & 241.4465 \\ 0 & 0 & 1 \end{bmatrix}$$

Extrinsic Matrix $[R|t]$

For each rotation vector and translation vector pair, you need to:

1. **Convert the Rotation Vector to a Rotation Matrix** R: Use Rodrigues' rotation formula or a library function like `cv2.Rodrigues` in OpenCV.

2. **Form the Extrinsic Matrix**: Combine the rotation matrix R and the translation vector t into a 4x4 matrix.

Example for a Single Extrinsic Parameter Set
Given:

- Rotation vector: $[-0.0010, 0.0871, 3.1164]$
- Translation vector (in millimeters): $[104.2676, 61.9279, 254.7238]$

Result Interpretation:
The intrinsic matrix K will remain as:

$$K = \begin{bmatrix} 504.3079 & 0 & 320.6196 \\ 0 & 505.2568 & 241.4465 \\ 0 & 0 & 1 \end{bmatrix}$$

For the extrinsic matrix, it will be a 4x4 matrix of the form:

$$[E] = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where R is the 3x3 rotation matrix obtained from Rodrigues' formula and t_x, t_y, t_z are the components of the translation vector.
This process can be repeated for each set of rotation and translation vectors to obtain their respective extrinsic matrices.
Since we need to know the coordinate of the object referring to the robotic arm base we multiply the found extrinsic matrix with the matrix that represents the position of the camera referring to the base and this reference is the world coordinate reference. The matrix is:

$$[T] = \begin{bmatrix} 0 & 0 & 1 & 92.5 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 139 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

General Conclusion

The research project focused on the design, development, and control of a mobile robot equipped with a manipulator arm and mecanum wheels, yielding several significant outcomes across its various chapters.
Chapter 1: General Overview of Robotics The research successfully provided a comprehensive understanding of the evolution of robotics. It highlighted how robots have evolved from performing repetitive tasks in industrial settings to more dynamic and intelligent applications such as mobile manipulation. The review of various robotic structures demonstrated the versatility and potential of robots in modern applications.
Chapter 2: System Design A key result from this chapter was the successful creation of detailed 3D models for the robotic system components. The design process allowed the team to simulate the robots movements and refine the mechanics before the physical assembly, ensuring smoother functionality. This simulation stage was crucial for minimizing errors during the physical build phase.
Chapter 3: Components of the Robot This chapter's major result was finding the most suitable components like motors, control boards, and sensors, which were essential for the robots functionality. By selecting appropriate types of motors and sensors, the project achieved a high level of control and responsiveness in the robot's movements and decision-making processes. The successful integration of these components formed the backbone of the robots operational capabilities.
Chapter 4: Assembly of the Robot The assembly process revealed several practical challenges, such as aligning electronic and mechanical parts, but these were overcome to create a functional system. The successful assembly of the robot demonstrated the project's capability to translate theoretical designs into a working prototype, with all components interacting seamlessly.
Chapter 5: Robot Manipulator Control A key result here was the implementation and fine-tuning of the PID controller, which significantly improved the robots movement precision and stability. The research showed how the PID controller effectively managed the robots manipulator arm and mecanum wheel movement, ensuring smooth motion and precise task execution. The chapter also showcased how control challenges, such as yaw control, were addressed through sensor integration, resulting in a stable and responsive robot.

Chapter 6: Camera Vision The research achieved successful integration of camera vision systems into the robot, enhancing its ability to perceive and respond to its environment. The calibration and processing of visual data allowed the robot to perform tasks like navigation and object detection with greater accuracy. This outcome demonstrated the potential of combining camera vision with robotic systems for more intelligent and autonomous behaviors.

Overall, the project achieved its goal of designing a functional robotic system with advanced control mechanisms. The use of 3D modeling, sensor integration, and control algorithms like PID highlighted the projects ability to combine both theoretical knowledge and practical application. Each chapter contributed valuable insights into robotics, from design to real-time control, with promising results for future development in robotic systems capable of complex tasks.

Bibliography

[1] International Federation of Robotics. World robotics 2023 report: Asia ahead of europe and the americas, 2023.

[2] Liwei Yang, Ping Li, Song Qian, He Quan, Jinchao Miao, Mengqi Liu, Yanpei Hu, and Erexidin Memetimin. Path planning technique for mobile robots: A review. *Machines*, 11(10):980, 2023.

[3] Lorenzo Sciavicco and Bruno Siciliano. *Modelling and control of robot manipulators*. Springer Science & Business Media, 2012.

[4] Arduino Store. Tinkerkit braccio robot.

[5] Dejan Nedelkovski. Howtomechatronics: Tutorials, projects, and resources on electronics, robotics, and automation.

[6] Jordi Palacín and David Martínez. Improving the angular velocity measured with a low-cost magnetic rotary encoder attached to a brushed dc motor by compensating magnet and hall-effect sensor misalignments. *Sensors*, 20(3):678, 2020.

[7] Hang Lu, Qing Li, and Hongwei Zhao. Pid control design for dc motor speed control based on matlab. In *2017 36th Chinese Control Conference (CCC)*, pages 4746–4750. IEEE, 2017.

[8] Luiz A. Santana, Felipe R. Silva, and Leonardo N. Castoldi. Physical modeling and parameters identification of the mg995 servomotor. *ResearchGate*, 2021.

[9] Last Minute Engineers. Drv8833 dual motor driver module arduino tutorial, 2023.

[10] Last Minute Engineers. Mpu6050 accelerometer + gyro tutorial for arduino, 2023.

[11] D. Joseph Mathew. *Beginning Robotics with Raspberry Pi and Arduino: Using Python and OpenCV*. Apress, 2021.

[12] Yuanfei Xu, Yuwei Zhao, and ShiLong Zhao. Grid line tracking omnidirectional robot based on visible light sensor and mecanum wheel pid control. In *2021 7th International Symposium on Mechatronics and Industrial Informatics (ISMII)*, pages 57–60, 2021.

[13] J. Kim and Y. Kim. Design of a pid controller for an automatic voltage regulator using a genetic algorithm. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pages 3545–3550, Hawaii, USA, October 2005. IEEE.

[14] F. J. Gonzalez and R. Zbikowski. Optimization of control laws for flapping-wing micro air vehicles. In *Proceedings of the IEEE International Conference on Control Applications*, pages 837–842, Glasgow, UK, September 2002. IEEE.

[15] Elprocus. The working of a pid controller, 2023.

[16] Abdullah Al-Qasabi Ahmed S. Hussein and Ramzi Shakhatreh. Design, implementation, and digital control of a robotic arm. *International Journal of Robotics*, 2013.

[17] X. Deng, H. Gao, and J. Li. An advanced robot control system for service robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1005–1010, Barcelona, Spain, April 2005. IEEE.

[18] K.J. Åström and T. Hägglund. *PID Controllers: Theory, Design, and Tuning*. ISA-The Instrumentation, Systems, and Automation Society, 2nd edition edition, 2002.

[19] IBM. What is interrupt processing?

[20] Hari Wibawa, Oyas Wahyunggoro, and Adha Imam Cahyadi. Dc motor speed control using hybrid pid-fuzzy with itae polynomial initiation. *IJITEE (International Journal of Information Technology and Electrical Engineering)*, 3(1):7–15, 2019.

[21] Sri Nurhasanah, Irwan Prihatin, and Asep Hamdani Azis. Embedded control system of dc motor using microcontroller arduino and pid algorithm. *Journal of Physics: Conference Series*, 1933(1):012006, 2021.

[22] Hui-min Li, Xiao-bo Wang, Shang-bin Song, and Hao Li. Vehicle control strategies analysis based on pid and fuzzy logic control. *Procedia engineering*, 137:234–243, 2016.

[23] L. M. Saoud, I. Stiharu, M. E. Kamas, and H. C. So. Neuro-adaptive control of a manipulator arm. *IEEE Access*, 8:21346–21355, 2020.

[24] Naveen Gupta and Ashwani Kumar. Comparing different control strategies for position control of dc servo motor. In *2011 International Conference on Emerging Trends in Electrical and Computer Technology*, pages 15–19. IEEE, 2011.

[25] Zhiqiang Zhou, Haibin Lin, Xiaoqian Chen, and Zhenfeng Qiu. Control theory and applications for engineering: An introduction. *IET The Journal of Engineering*, 2021(70006):1–10, 2021.

[26] P K Padhy, Takeshi Sasaki, Sousuke Nakamura, and Hideki Hashimoto. Modeling and position control of mobile robot. In *2010 11th IEEE International Workshop on Advanced Motion Control (AMC)*, pages 100–105, 2010.

[27] Arduino programming with matlab and simulink. https://www.mathworks.com/discovery/arduino-programming-matlab-simulink.html, 2024.

[28] Muhammad Zakwan Bin Abdul Karim and Norashikin M Thamrin. Servo motor controller using pid and graphical user interface on raspberry pi for robotic arm. In *Journal of Physics: Conference Series*, volume 2319, page 012015. IOP Publishing, 2022.

[29] Waluyo Adi Siswanto, Muhamad Riza, and Asep Ridwan Budijuwono. Alternative control system for robot arm with data logger. *International Journal of Mechatronics and Automation*, 2020.

[30] MathWorks. Camera calibration - vision toolbox, 2023.

[31] Towards Data Science. What are intrinsic and extrinsic camera parameters in computer vision, 2023.

[32] Baeldung. Focal length and intrinsic camera parameters, 2023.

[33] Yan Cao, Jianghao Fu, and Yu Bai. Camera calibration and error analysis based on matlab calibration tool. *School of Mechatronic Engineering, Xi'an Technological University, Xi'an 710032, China*, 2021.